9.3

Vývoj aplikací pro IBM MQ

Poznámka

Než začnete používat tyto informace a produkt, který podporují, přečtěte si informace, které uvádí “Poznámky” na stránce 1253.

Toto vydání se vztahuje na verzi 9 vydání 3 produktu IBM® MQ a na všechna následná vydání a úpravy, není-li v nových vydáních uvedeno jinak.

Obsah

Vyvíjení aplikací pro IBM MQ

Můžete vyvíjet aplikace pro odesílání a příjem zpráv a pro správu správců front a souvisejících prostředků. Produkt IBM MQ podporuje aplikace napsané v mnoha různých jazycích a rámcích.

Začínajíte s vývojem aplikací pro produkt IBM MQ?

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

- IBM MQ Essentials pro vývojáře *(naučit se základy, spustit ukázku, kódovat aplikaci, provést pokročilejší výukové programy)*
- IBM MQ Soubory ke stažení pro vývojáře *(včetně bezplatných verzí pro vývojáře a zkušebních verzí)*

Můžete také snáze vyvíjet aplikace, pokud jste obeznámeni s koncepcemi popsanými v následujících sekcích:

- “Koncepty vývoje aplikací” na stránce 7
- “Aspekty návrhu pro aplikace IBM MQ” na stránce 47

Podpora objektově orientovaných jazyků a rámců

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

- JMS
- Java
- C++
- .NET

Další informace najdete v tématu “Objektově orientované aplikace” na stránce 15.

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

- C# ukázkový kód a ukázkové aplikace
- Ukázkové aplikace C++
- Visual Basic ukázkové aplikace

Viz “Psaní a nasazení programů IBM MQ .NET” na stránce 563.

Produkt IBM MQ podporuje jádro .NET pro aplikace v prostředích Windows z produktu IBM MQ 9.1.1 a pro aplikace v prostředích Linux® z produktu IBM MQ 9.1.2. Další informace viz “Instalace produktu IBM MQ classes for .NET” na stránce 541.

ALW Produkt IBM MQ také podporuje klienty AMQP, kteří implementují protokol OASIS AMQP 1.0 .

MQ Light, Apache Klienti Qpid jako Apache Qpid Proton a Apache Qpid JMS API jsou založeny na tomto protokolu.

Rozhraní API MQ Light jsou k dispozici na adrese IBM MQ Light.

Klienti Apache Qpid jsou k dispozici na adrese QPid Proton.

Následující jazykové vazby jsou poskytovány tak, jak jsou:

- a Přejít na vazbu
- implementace rozhraní API JavaScript , která pracuje s aplikacemi Node.js

Podpora programových rozhraní REST API

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

- IBM MQ messaging REST API
- z/OS IBM z/OS Připojit EE
- IBM Integration Bus
- IBM DataPower Brána

Viz téma “Vývoj aplikací REST pomocí produktu IBM MQ” na stránce 682a také výukový program Začínáme s rozhraním REST API systému zpráv IBM MQ v oblasti IBM MQ produktu IBM Developer. Tento výukový program obsahuje příklady v následujících jazycích, které jsou poskytovány tak, jak jsou, pro použití s produktem IBM MQ messaging REST API:

- Příklad Jdi, který používá 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:

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

Tyto jazyky používají rozhraní fronty zpráv (MQI) pro přístup ke službám front zpráv. Viz “Vývoj aplikací MQI pomocí produktu IBM MQ” na stránce 696. Všimněte si, že objektový model IBM MQ , používaný objektově orientovanými jazyky a rámci, poskytuje další funkce, které nejsou k dispozici procedurálním jazykům pomocí rozhraní MQI.

Určení názvu aplikace

ALW

Před IBM MQ 9.1.2můžete zadat název aplikace v Java nebo JMS klientských aplikacích. V produktu IBM MQ 9.1.2můžete také zadat název aplikace v dalších programovacích jazycích. Další informace viz “Určení názvu aplikace v podporovaných programovacích jazycích” na stránce 50.

Související úlohy

“Vyvíjení aplikací pro MQ Telemetry” na stránce 1200

“Vývoj aplikací Microsoft Windows Communication Foundation pomocí IBM MQ” na stránce 1222
Vlastní kanál produktu Microsoft Windows Communication Foundation (WCF) pro produkt IBM MQ odesílá a přijímá zprávy mezi klienty a službami WCF.

Související odkazy

“Vyvíjení aplikací pro Managed File Transfer” na stránce 1179
Uveďte programy, které se mají spustit s produktem Managed File Transfer, použijte Apache Ant s Managed File Transfer, přizpůsobte Managed File Transfer s uživatelskými uživatelskými programy a řídte Managed File Transfer vložením zpráv do fronty příkazů agenta.

Koncepty vývoje aplikací

K psaní aplikací IBM MQ můžete použít výběr procedurálních nebo objektově orientovaných jazyků. Než začnete navrhovat a psát aplikace IBM MQ , seznamte se se základními koncepty produktu IBM MQ .

Informace o typech aplikací, které můžete napsat pro IBM MQ, viz “Vyvíjení aplikací pro IBM MQ” na stránce 5 a “Akce, které mohou aplikace provádět” na stránce 7.

Související pojmy

“Aspekty návrhu pro aplikace IBM MQ” na stránce 47
Když jste se rozhodli, jak mohou vaše aplikace využívat platformy a prostředí, která máte k dispozici, musíte se rozhodnout, jak používat funkce nabízené produktem IBM MQ.

Akce, které mohou aplikace provádět

Můžete vyvíjet aplikace pro odesílání a příjem zpráv, které potřebujete pro podporu svých obchodních procesů. Můžete také vyvíjet aplikace pro správu správců front a souvisejících prostředků.

Akce, které mohou aplikace provádět na systému IBM MQ for Multiplatforms

V systému Multiplatformsmůžete psát aplikace, které provádějí následující akce:

- Odesílat zprávy do jiných aplikací spuštěných ve stejných operačních systémech. Aplikace mohou být buď na stejném, nebo na jiném systému.
- Odesílat zprávy aplikacím, které jsou spuštěny na jiných platformách IBM MQ .
- Použití řazení zpráv do fronty z produktu CICS pro následující systémy:
  - 

položky TXSeries pro AIX
  - 

IBM i
  - 

Windows
- Použití řazení zpráv z Encina do fronty pro následující systémy:
  - 

AIX
  - 

Windows
- Použití řazení zpráv do fronty z produktu Tuxedo pro následující systémy:
  - 

AIX
  - AT & T
  - 

Windows
- Použijte produkt IBM MQ jako správce transakcí, který koordinuje aktualizace provedené externími správci prostředků v rámci IBM MQ pracovních jednotek. Následující externí správci prostředků jsou podporováni a jsou v souladu s rozhraním XA sdružení X/OPEN.
  - Db2
  - Informix
  - Oracle
  - Sybase
- Zpracujte několik zpráv společně jako jednu pracovní jednotku, kterou lze potvrdit nebo vycouvat.
- Spusťte z úplného prostředí IBM MQ nebo z klientského prostředí IBM MQ .

Akce, které mohou aplikace provádět na systému IBM MQ for z/OS

z/OS

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

- Použijte řazení zpráv do fronty v rámci CICS nebo IMS.
- Odesílejte zprávy mezi dávkovými aplikacemi, aplikacemi CICSa IMS a vyberte nejvhodnější prostředí pro každou funkci.
- Odesílat zprávy aplikacím, které jsou spuštěny na jiných platformách IBM MQ .
- Zpracujte několik zpráv společně jako jednu pracovní jednotku, kterou lze potvrdit nebo vycouvat.
- Odesílat zprávy do aplikací IMS a pracovat s nimi prostřednictvím mostu IMS .
- Podílet se na jednotkách práce koordinovaných RRS.

Každé prostředí v rámci produktu z/OS má své vlastní charakteristiky, výhody a nevýhody. Výhodou produktu IBM MQ for z/OS je, že aplikace nejsou vázány na žádné prostředí, ale mohou být distribuovány tak, aby využívaly výhod jednotlivých prostředí. Můžete například vyvíjet rozhraní koncových uživatelů pomocí TSO nebo CICS, spouštět moduly náročné na zpracování v dávce z/OS a spouštět databázové aplikace v produktu IMS nebo CICS. Ve všech případech mohou různé části aplikace komunikovat pomocí zpráv a front.

Návrháři aplikací IBM MQ si musí být vědomi rozdílů a omezení, která tato prostředí vyžadují. Příklad:

- Produkt IBM MQ poskytuje prostředky, které umožňují komunikaci mezi správci front (označované jako *distribuované řazení do front*).
- Metody potvrzování a zálohování změn se liší mezi dávkovým prostředím a prostředím CICS .
- Produkt IBM MQ for z/OS poskytuje podporu v prostředí IMS pro online programy pro zpracování zpráv (MPP), interaktivní programy rychlé cesty (IFP) a programy pro dávkové zpracování zpráv (BMP). Pokud píšete dávkové programy DL/I, postupujte podle pokynů uvedených v tématech jako “Sestavení dávkových aplikací z/OS” na stránce 990 a “z/OS aspekty dávkového zpracování” na stránce 710 pro dávkové programy z/OS .
- Ačkoli v jednom systému z/OS může existovat více instancí produktu IBM MQ for z/OS , oblast CICS se může v daném okamžiku připojit pouze k jednomu správci front. Ke stejnému správci front však může být připojena více než jedna oblast CICS . V dávkovém prostředí 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, což zvyšuje propustnost a dostupnost. Takové 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 zadáním názvu skupiny sdílení front namísto konkrétního názvu správce front. Toto je známé jako *skupinové dávkové připojení*, nebo jednodušeji *skupinové připojení*. Viz Sdílené fronty a skupiny sdílení front.

z/OS Rozdíly mezi podporovanými prostředími a jejich omezeními jsou vysvětleny v části “Používání a psaní aplikací na systému IBM MQ for z/OS” na stránce 858.

Související pojmy

“Koncepty vývoje aplikací” na stránce 7
K psaní aplikací IBM MQ můžete použít výběr procedurálních nebo objektově orientovaných jazyků. Než začnete navrhovat a psát aplikace IBM MQ , seznamte se se základními koncepty produktu IBM MQ .

“Aspekty návrhu pro aplikace IBM MQ” na stránce 47
Když jste se rozhodli, jak mohou vaše aplikace využívat platformy a prostředí, která máte k dispozici, musíte se rozhodnout, jak používat funkce nabízené produktem IBM MQ.

“Psaní procedurální žádosti o zařazení do fronty” na stránce 699
Prostřednictvím těchto informací můžete získat informace o psaní aplikací řazení do front, o připojování a odpojování od správce front, o publikování/odběru a o otevírání a zavírání objektů.

“Psaní procedurálních aplikací klienta” na stránce 882

Co potřebujete vědět, abyste mohli psát klientské aplikace na systému IBM MQ pomocí procedurálního jazyka.

“Použití IBM MQ classes for JMS/Jakarta Messaging” na stránce 78
IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging jsou poskytovatelé systému zpráv Java dodávaní s produktem IBM MQ. Kromě implementace rozhraní definovaných ve specifikacích JMS a Jakarta Messaging tito poskytovatelé systému zpráv přidávají do rozhraní API systému zpráv Java dvě sady rozšíření.

“Použití produktu IBM MQ classes for Java” na stránce 335
Použijte IBM MQ v prostředí Java . IBM MQ classes for Java Povolit aplikaci Java připojit se k produktu IBM MQ jako klient IBM MQ nebo se připojit přímo ke správci front IBM MQ .

“Vývoj aplikací .NET” na stránce 539
IBM MQ classes for .NET Povolit aplikacím .NET připojit se k produktu IBM MQ jako k serveru IBM MQ MQI client nebo se připojit přímo k serveru IBM MQ .

“Vývoj aplikací C++” na stránce 512
Produkt IBM MQ poskytuje třídy C++ ekvivalentní objektům IBM MQ a některé další třídy ekvivalentní datovým typům pole. Poskytuje řadu funkcí, které nejsou k dispozici prostřednictvím rozhraní MQI.

“Sestavení procedurální aplikace” na stránce 967
Aplikaci IBM MQ můžete napsat v jednom z několika procedurálních jazyků a spustit ji na několika různých platformách.

Související úlohy

“Použití ukázkových procedurálních programů IBM MQ” na stránce 1023
Tyto ukázkové programy jsou napsány v procedurálních jazycích a ukazují typické použití rozhraní MQI (Message Queue Interface). Programy IBM MQ na různých platformách.

“Vývoj aplikací Microsoft Windows Communication Foundation pomocí IBM MQ” na stránce 1222
Vlastní kanál produktu Microsoft Windows Communication Foundation (WCF) pro produkt IBM MQ odesílá a přijímá zprávy mezi klienty a službami WCF.

Zabezpečení

Multi Aplikace, názvy aplikací a instance aplikací

Než začnete navrhovat a psát své aplikace, seznamte se se základními koncepty aplikací, názvy aplikací a instancemi aplikací.

Aplikace

Multi

Připojení ke správci front jsou považována za pocházející ze stejné *aplikace* , pokud poskytují stejný *název aplikace*. Název aplikace se zobrazí jako atribut APPLTAG příkazu DISPLAY CONN (*) TYPE CONN.

Notes:

1. V případě aplikací, které používají verzi IBM MQ client dřívější než IBM MQ 9.1.2, je název aplikace automaticky nastaven pomocí IBM MQ client. Jeho hodnota závisí na programovacím jazyku aplikace a na platformě, na které je aplikace spuštěna. Další informace viz PutApplNázev .
2. V případě aplikací IBM MQ client používajících IBM MQ client at IBM MQ 9.1.2 nebo novější je možné nastavit název aplikace na specifickou hodnotu. Ve většině případů to nevyžaduje změny kódu aplikace ani nutnost znovu zkompilovat aplikaci. Další informace viz “Použití názvu aplikace v podporovaných programovacích jazycích” na stránce 51.

Instance aplikace

Multi

Připojení jsou dále rozdělena do *instancí aplikace*. Instance aplikace je sada úzce souvisejících připojení, která poskytují jednu 'jednotku provedení' pro tuto aplikaci. Obvykle se jedná o jeden proces operačního systému, který může mít několik podprocesů a přidružená připojení produktu IBM MQ .

V systému IBM MQ for Multiplatforms je instance aplikace přidružena ke specifické značce připojení. Správce front automaticky přidruží nová připojení k existující instanci aplikace, pokud uvidí, že spolu souvisejí.

Notes:

- Používáte-li připojení klienta, mohou se tyto procesy připojit ke správci front prostřednictvím jednoho nebo více spuštěných kanálů.
- V aplikacích JMS se instance aplikace mapuje na specifické připojení JMS a všechny přidružené relace JMS .

Instance aplikací jsou obzvláště důležité v systému IBM MQ for Multiplatforms při použití automatického vyrovnávání aplikacíjednotného klastru. Na platformách IBM MQ for Multiplatforms můžete aktuálně připojené instance aplikací zobrazit pomocí příkazu DISPLAY APSTATUS .

V některých případech nemůže správce front správně provést připojení k přidružení instance aplikace, zejména:

- Je-li provedeno více připojení ke sdílené konverzaci ze stejného procesu s použitím různých názvů aplikací.
- Pokud se používají knihovny klienta starší úrovně. Například instalace klienta produktu IBM MQ JMS na adrese IBM MQ 9.1.2 a dřívější.

V těchto situacích, pokud se aplikace nedefinují jako opětovně připojitelné, bude to povoleno, ale některé skupiny instancí aplikací mohou být nesprávné. Je-li některé z připojení deklarováno jako MQCNO_RECONNECT, má to výrazně negativní vliv na vyvážení aplikací; volání MQCONN bude proto odmítnuto s hodnotou MQCNO_RECONNECT_NEKOMPATIBILNÍ.

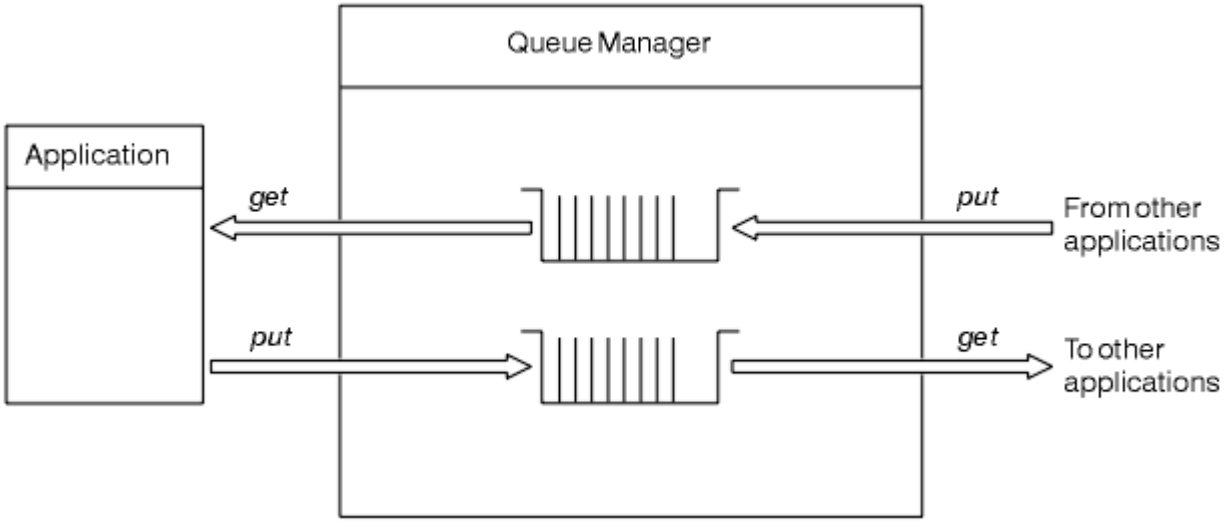
Související pojmy

“Určení názvu aplikace v podporovaných programovacích jazycích” na stránce 50
Před produktem IBM MQ 9.2.0již můžete zadat název aplikace v klientských aplikacích typu Java nebo JMS . Od produktu IBM MQ 9.2.0 je tato funkce rozšířena na další programovací jazyky v systému IBM MQ for Multiplatforms.

Aplikační programy používající rozhraní MQI

Aplikační programy IBM MQ potřebují určité objekty, aby je bylo možné úspěšně spustit.

Produkt Obrázek 1 na stránce 11 zobrazuje aplikaci, která odebírá zprávy z fronty, zpracovává je a poté odesílá 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 získávat pouze přímo z lokálních front (pomocí příkazu MQGET).

Před spuštěním této aplikace musí být splněny následující podmínky:

- Správce front musí existovat a být spuštěn.
- Musí být definována první fronta aplikací, ze které mají být zprávy odebrány.
- Musí být také definována druhá fronta, do které aplikace vkládá zprávy.
- Aplikace se musí být schopna připojit ke správci front. Chcete-li to provést, musí být propojen s IBM MQ. Viz “Sestavení procedurální aplikace” na stránce 967.
- Aplikace, které vkládají zprávy do první fronty, se musí také připojit ke správci front. Pokud jsou vzdálené, musí být také nastaveny s přenosovými frontami a kanály. Tato část systému není v souboru Obrázek 1 na stránce 11zobrazena.

Použití připojení klienta pro připojení k více správcům front IBM MQ

Je možné konfigurovat aplikace připojené ke klientovi tak, aby se připojovaly k více než jednomu správci front (kvůli vyrovnání zátěže nebo dostupnosti služeb).

Primární mechanizmy k dosažení tohoto cíle v klientu IBM MQ jsou použití tabulek definic kanálů klienta, viz Konfigurace tabulek definic kanálů klientanebo seznamy připojení.

Je také možné dosáhnout podobného chování pomocí externích produktů pro vyrovnávání zátěže nebo zabalením kódu připojení IBM MQ do 'stubu', který může přesměrovat názvy hostitelů nebo adresy IP.

Každá z těchto technik je vybavena určitými omezeními a může být více či méně vhodná pro konkrétní požadavky aplikace. Následující části, i když nejsou vyčerpávající, popisují konkrétní aspekty, které byste měli zvážit, a vliv těchto různých přístupů na tyto aspekty.

IBM MQ uniformní klastry, viz O uniformních klastrech, poskytují výkonný mechanizmus pro dosažení vodorovného škálování aplikací ve více správcích front, které staví na základním mechanizmu tabulky CCDT a poskytují více cílů. Uniformní klastry mohou poskytovat schopnosti nad rámec toho, co je možné pomocí externího prostředku pro vyrovnávání zátěže, který nemá informace o základních protokolech IBM MQ , a vyhnout se některým problémům uvedeným níže, proto zvažte použití jednotného klastru přednostně před jinými technikami, kde je to vhodné.

Upozornění: Aplikace používající produkt IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging, včetně aplikací využívajících jeden z adaptérů prostředků IBM MQ , které se připojují ke správcům front pomocí technologií pro vyrovnávání zátěže, byste měli používat s opatrností. Pokud se setkáte s problémy, znovu tyto problémy vytvořte bez pokusu o použití vyrovnání zátěže.

Jedná se o několik otázek, z nichž všechny znamenají, že taková spojení jsou v nejlepším případě problematická a v nejhorším případě zcela nespolehlivá:

- Při připojování libovolné aplikace, která vytváří více připojení ke správci front s použitím libovolné formy vyrovnávání zátěže, je třeba věnovat zvláštní pozornost. To zahrnuje všechny aplikace používající IBM MQ Třídy pro JMS/Jakarta Messaging , protože vytvářejí více IBM MQ připojení v obecném použití. Používáte-li externí prostředek pro vyrovnávání zátěže nebo stub vlastního kódu, musí tento prostředek vždy směrovat připojení ze stejné instance aplikace do stejného správce front.
- Použití správy transakcí XA nebo rozhraní JTA (Java Transaction API) spoléhá na schopnost konzistentního připojení ke stejnému správci front-v praxi je nepravděpodobné, že by to bylo někdy praktické s jakýmkoli způsobem vyrovnávání zátěže.
- -Jednotná správa klastrů spoléhá na schopnost instruovat klienty k opětovnému připojení ke specifickým správcům front bez zásahů. Nedoporučuje se pokoušet se kombinovat externí vyvažování zátěže s použitím jednotných klastrů.

Chcete-li dosáhnout vodorovného škálování aplikací ve více správcích front, měli byste použít funkci jednotného klastru IBM MQ , nikoli technologie externího vyrovnávání zátěže. Informace o uniformních klastrech včetně způsobu vytváření a používání uniformních klastrů naleznete v tématu Konfigurace uniformního klastrua v následujících tématech.

Termíny použité v těchto informacích

CCDT-multi-QMGR
Znamená soubor CCDT, který obsahuje více kanálů připojení klienta (CLNTCONN) se stejnou skupinou, což je atribut připojení klienta názvu správce front (QMNAME CLNTCONN), kde se různé položky CLNTCONN interpretují na různé správce front.

Liší se od souboru CCDT, který obsahuje více položek CLNTCONN, které jsou jednoduše různé adresy IP nebo názvy hostitelů pro stejného správce front s více instancemi, což je přístup, který můžete zvolit pro kombinaci se stubem kódu.

Pokud zvolíte přístup k více správcům front CCDT, je třeba zvolit, zda mají být položky seřazeny podle priority, nebo zda má být nastavena náhodná správa pracovní zátěže (WLM):

S nastavenou prioritou
Použijte více abecedně řazených záznamů s atributy CLNTWGHT (1) a AFFINITY (PREFERRED), abyste si zapamatovali poslední dobré připojení.

Náhodně
Použijte atributy CLNTWGHT (1) a AFFINITY (NONE). Můžete upravit váhu WLM na různých serverech IBM MQ úpravou CLNTWGHT.

Poznámka: Měli byste se vyhnout velkým rozdílům v CLNTWGHT mezi kanály.

Prostředek pro vyrovnávání zátěže
Znamená síťové zařízení s virtuální adresou IP (VIP) konfigurovanou s monitorováním portů modulů listener protokolu TCP/IP více správců front produktu IBM MQ . Způsob konfigurace VIP v síťovém zařízení závisí na síťovém zařízení, které používáte.

Následující volby se týkají pouze aplikací odesílajících zprávy nebo zahajujících synchronní zasílání zpráv požadavků a odpovědí. Aspekty pro aplikace obsluhující tyto zprávy a požadavky, například moduly listener, jsou zcela oddělené a jsou podrobně popsány v tématu "Připojení modulu listener pro zprávy k frontě".

Změna rozsahu kódu vyžadovaná pro existující aplikace, které se připojují k jednomu správci front

Seznam CONNAME, CCDT multi-QMGR a prostředek pro vyrovnávání zátěže
MQCONN ("QMNAME") do MQCONN ("*QMNAME")

Název správce front může být v konfiguraci rozhraní JNDI (Java Naming and Directory Interface) pro aplikace Java Platform, Enterprise Edition (Java EE). Jinak to vyžaduje jednoznakovou změnu kódu.

Stub kódu
Nahraďte existující logiku připojení JMS nebo MQI stubem kódu.

Podpora různých strategií WLM

Seznam CONNAME
Pouze s prioritou.

To pravděpodobně bude mít negativní vliv na kód.

CCDT multi-QMGR
Priorita nebo náhodná.

To pravděpodobně nebude mít žádný vliv na kód.

Prostředek pro vyrovnávání zátěže
Libovolná, včetně každého připojení pro všechny zprávy.

To pravděpodobně bude mít pozitivní vliv na kód.

Stub kódu
Libovolná, včetně každé zprávy pro všechny zprávy.

To pravděpodobně bude mít pozitivní vliv na kód.

Režie výkonu při nedostupnosti primárního správce front

Seznam CONNAME
Vždy se pokusí o první v seznamu.

To pravděpodobně bude mít negativní vliv na kód.

CCDT multi-QMGR
Pamatuje si poslední dobré spojení.

To pravděpodobně bude mít pozitivní vliv na kód.

Prostředek pro vyrovnávání zátěže
Monitorování portů se vyhýbá nesprávným správcům front.

To pravděpodobně bude mít pozitivní vliv na kód.

Stub kódu
Dokáže si vzpomenout na poslední dobré spojení a inteligentně to zkusit.

To pravděpodobně bude mít pozitivní vliv na kód.

Podpora transakcí XA

Seznam CONNAME, CCDT multi-QMGR a prostředek pro vyrovnávání zátěže
Správce transakcí musí ukládat informace o zotavení, které se znovu připojí ke stejnému prostředku správce front.

Volání MQCONN, které se interpretuje pro různé správce front, toto obecně zneplatňuje. Například v prostředí Java EE by se měla jedna továrna připojení při použití XA převést na jednoho správce front.

To pravděpodobně bude mít negativní vliv na kód.

Stub kódu
Stub kódu může splňovat požadavky XA pro správce transakcí, například více továren připojení.

To pravděpodobně bude mít pozitivní vliv na kód.

Administrativní flexibilita pro skrytí změn infrastruktury z aplikací

Seznam CONNAME
Pouze DNS.

To pravděpodobně bude mít negativní vliv na kód.

CCDT multi-QMGR
DNS a sdílený systém souborů nebo sdílený systém souborů nebo odeslání typu push souboru CCDT.

To pravděpodobně nebude mít žádný vliv na kód.

Prostředek pro vyrovnávání zátěže
Dynamická virtuální IP adresa (VIP).

To pravděpodobně bude mít pozitivní vliv na kód.

Stub kódu
DNS nebo jednotlivé položky CCDT správce front.

To pravděpodobně nebude mít žádný vliv na kód.

Zamezení narušení plánované údržby

Existuje další situace, kterou je třeba zvážit a naplánovat, což je způsob, jak se vyhnout narušení aplikací, například chyb a vypršení časového limitu viditelných pro koncové uživatele, během plánované údržby správce front. Nejlepším způsobem, jak se vyhnout přerušení, je před zastavením odebrat veškerou práci ze správce front.

Zvažte scénář požadavku a odpovědi. Chcete dokončit všechny probíhající požadavky a odpovědi, které má aplikace zpracovat, ale nechcete, aby byla do systému odeslána žádná další práce. Pouhé uvedení správce front do klidového stavu tuto potřebu nesplňuje, protože dobře kódované aplikace obdrží návratový kód RC2161 Výjimka MQRC_Q_MGR_QUIESCING, než obdrží své zprávy odpovědi pro probíhající požadavky.

Můžete nastavit PUT (DISABLED) ve frontách požadavků použitých k odeslání práce, zatímco ponecháte fronty odpovědí jak PUT (ENABLED), tak GET (ENABLED). Tímto způsobem můžete monitorovat hloubku front požadavků, přenosů a odpovědí. Jakmile se všechny stabilizují, tj. požadavky za letu se dokončí nebo jim dojde časový limit, můžete zastavit správce front.

Pro zpracování fronty požadavků PUT (DISABLED) je však vyžadováno správné kódování v požadujících aplikacích, což při pokusu o odeslání zprávy vede k chybě MQRC_PUT_INHIBITED (návratový kód RC2051).

Všimněte si, že k výjimce nedojde při vytváření připojení k produktu IBM MQnebo při otevírání fronty požadavků. K výjimce dochází pouze v případě, že dojde k pokusu o skutečné odeslání zprávy pomocí volání MQPUT.

Sestavení stubu kódu, který zahrnuje tuto logiku ošetření chyb pro scénáře požadavků a odpovědí a požádá vaše aplikační týmy o budoucí použití takového stubu kódu, vám může pomoci vyvíjet aplikace s konzistentním chováním.

V 9.3.0 Vývoj flexibilních a rozšiřitelných klientských aplikací

V zájmu odolnosti proti poruchám a rozšiřitelnosti umožňuje implementace klientských aplikací, které podporují volby připojení do uniformních klastrů, znovu vyvážit instance aplikace mezi správci front.

Přehled uniformních klastrů viz O uniformních klastrech .

V ideálním případě je toto nové vyvážení pro aplikaci neviditelné, ale pro tento typ implementace jsou vhodné pouze určité typy aplikací a v návrhu aplikace může být zapotřebí určitá pozornost.

Tyto úvahy spadají do dvou hlavních kategorií:

- Vzácné *cesty k chybám* , které již mohou existovat pro znovu připojitelné aplikace, ale jsou pravděpodobnější při implementaci do jednotného klastru. Například po opětovném připojení se vrátí zpět jakákoli pracovní jednotka průchodnosti a resetují se kurzory procházení. Může se jednat o vzácnou událost pro znovu připojitelnou aplikaci v jejím aktuálním prostředí, a proto není kód aplikace zpracován tak čistě, jak je to možné. Přezkoumání logiky aplikace, aby se zajistilo, že je pro takové situace k dispozici odpovídající zacházení, pomáhá vyhnout se neočekávaným problémům.

- *Afinity* ke konkrétnímu správci front. Pokud víte, že se aplikace musí vždy připojit zpět ke stejnému nebo specifickému správci front, měla by být aplikace nakonfigurována tak, aby se znovu připojovala k tomuto správci front, nebo by neměla mít povoleno připojení k tomuto správci front. Tyto afinity však mohou být dočasné, například čekání na zprávu odpovědi. V následující části je popsáno ovlivnění algoritmu vyvažování za účelem zohlednění těchto afinit z kódu aplikace. Další podrobnosti o těchto volbách a o tom, jak dosáhnout podobného přístupu prostřednictvím konfigurace namísto kódu aplikace, naleznete v tématu Vyrovnávání opětovného vyvážení aplikací v uniformních klastrech.

Ovlivňování voleb opětovného připojení v rozhraní MQI

Další informace o MQCNO_RECONNECT viz Volby opětovného připojení .

Pokud víte, že se aplikace musí vždy připojit zpět ke stejnému nebo specifickému správci front, měla by být konfigurována jako MQCNO_RECONNECT_Q_MGR nebo MQCNO_RECONNECT_DISABLED.

Ovlivnění algoritmu vyvažování v rozhraní MQI

Avšak možná budete chtít řídit nebo ovlivnit toto opětovné vyvážení chování, aby vyhovovalo potřebám specifických typů aplikací; například minimalizace přerušení v letových transakcích nebo zajištění toho, aby aplikace žadatele obdržely své odpovědi před přesunutím.

Určitá výchozí žádoucí chování jsou předpokládána a diskutována v tématu Opětovné vyvážení aplikací v uniformních klastrech. Můžete také ovlivnit chování specifických aplikací v době konfigurace nebo implementace prostřednictvím souboru client.ini , jak je popsáno v tomto tématu.

V jiných situacích může mít větší smysl, aby bylo chování při vyrovnávání a požadavky součástí logiky aplikace. V těchto případech lze produktu IBM MQ při připojování ke správci front prostřednictvím volání MQCONNX ve struktuře s názvem MQBNO (volby vyvažování) zadat stejnou relevantní charakteristiku aplikace.

Zadáte-li strukturu MQBNO, musí dodat všechny informace vyžadované produktem IBM MQ , aby bylo možné rozhodnout o tom, jak a kdy má být aplikace vyzvána k opětovnému připojení k jinému správci front.

Musíte dodat:

- **Type** aplikace
- **Timeout** , ve kterém je instance vyvážena bez ohledu na stav
- Jakékoli speciální **BalanceOptions**

Výjimkou je, že časový limit můžete v případě potřeby ponechat jako MQBNO_TIMEOUT_DEFAULT.
V tomto případě se časový limit interpretuje na libovolnou hodnotu v souboru client.ini , v aplikaci nebo v globálních sekcích, pokud jsou poskytnuty, a v případě selhání na základní výchozí hodnotu 10 sekund.

Podrobnosti o formátu této struktury viz MQBNO .

Další informace o aplikacích .NET naleznete v tématu Opětovné vyvážení aplikací v prostředí .NET .

Objektově orientované aplikace

Produkt IBM MQ poskytuje podporu pro JMS, Java, C + + a .NET. Tyto jazyky a rámce používají objektový model IBM MQ , který poskytuje třídy, které poskytují stejné funkce jako volání a struktury produktu IBM MQ .

Některé jazyky a rámce, které používají objektový model IBM MQ , poskytují další funkce, které nejsou k dispozici při použití procedurálních jazyků s rozhraním fronty zpráv (MQI).

Podrobnosti o třídách, metodách a vlastnostech poskytovaných tímto modelem viz “Objektový model IBM MQ” na stránce 16.

JMS

Produkt IBM MQ poskytuje třídy, které implementují specifikace Jakarta Messaging 3.0 a Java Message Service 2.0 . Podrobnosti o souboru IBM MQ classes for JMSviz Použití IBM MQ classes

for JMS. Chcete-li získat informace o rozdílech mezi IBM MQ classes for Java a IBM MQ classes for JMS, které vám pomohou rozhodnout, které použít, viz “Vývoj aplikací JMS/Jakarta Messaging a Java” na stránce 77.

IBM MQ Message Service Client (XMS) for C/C++ a IBM MQ Message Service Client (XMS) for .NET poskytují rozhraní API s názvem XMS , které má stejnou sadu rozhraní jako Java Message Service (JMS) Rozhraní API. Další informace viz téma “Vývoj aplikací XMS .NET” na stránce 598.

Java

Informace o kódování programů používajících IBM MQ objektový model v Javaviz Použití IBM MQ classes for Java .

Stabilized Produkt IBM neprovede žádná další vylepšení produktu IBM MQ classes for Java a jsou funkčně stabilizovány na úrovni dodané v produktu IBM MQ 8.0. Chcete-li získat informace o rozdílech mezi IBM MQ classes for Java a IBM MQ classes for JMS , které vám pomohou rozhodnout, které použít, viz “Vývoj aplikací JMS/Jakarta Messaging a Java” na stránce 77.

C++

Produkt IBM MQ poskytuje třídy C++ ekvivalentní objektům IBM MQ a některé další třídy ekvivalentní datovým typům pole. Poskytuje řadu funkcí, které nejsou k dispozici prostřednictvím rozhraní MQI. Informace o kódování programů používajících objektový model IBM MQ v jazyku C + +. Klienti služby systému zpráv pro jazyk C/C++ a .NET poskytují rozhraní API (Application Programming Interface) s názvem XMS , které má stejnou sadu rozhraní jako rozhraní Java Message Service (JMS), viz téma Použití C++ . Rozhraní API.

.NET

Informace o kódování programů .NET pomocí tříd IBM MQ .NET viz Vývoj aplikací .NET . Klienti služby zpráv pro C/C++ a .NET poskytují rozhraní API s názvem XMS , které má stejnou sadu rozhraní jako Java Message Service (JMS). Rozhraní API.

Související pojmy

“Vývoj aplikací MQI pomocí produktu IBM MQ” na stránce 696
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.

Technický přehled

“Koncepty vývoje aplikací” na stránce 7
K psaní aplikací IBM MQ můžete použít výběr procedurálních nebo objektově orientovaných jazyků. Než začnete navrhovat a psát aplikace IBM MQ , seznamte se se základními koncepty produktu IBM MQ .

Související odkazy

Odkaz na vývoj aplikací

Objektový model IBM MQ

Objektový model IBM MQ se skládá z tříd, metod a vlastností.

Objektový model IBM MQ se skládá z:

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

Při vytváření aplikace IBM MQ pomocí objektového modelu IBM MQ vytváříte v aplikaci instance těchto tříd. Instance třídy v objektově orientovaném programování se nazývá *objekt*. Po vytvoření objektu budete s objektem interagovat kontrolou nebo nastavením hodnot vlastností objektu (ekvivalent volání MQINQ nebo MQSET) a provedením volání metod pro objekt (ekvivalent volání jiných MQI).

Třídy

Model objektů IBM MQ poskytuje následující základní sadu tříd.

Skutečná implementace modelu se mezi různými podporovanými objektově orientovanými prostředími mírně liší.

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

MQQUEUE
Objekt třídy MQQueue představuje frontu. Má metody pro vložení () a získání () zpráv do fronty a z fronty (ekvivalent MQPUT a MQGET). Má vlastnosti odpovídající atributům fronty. Přístup k vlastnosti atributu fronty nebo zadání volání metody Put () nebo Get () implicitně otevře frontu (ekvivalent MQOPEN). Zničení objektu MQQueue implicitně zavře frontu (ekvivalent MQCLOSE).

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

Zpráva MQMessage
Objekt třídy MQMessage představuje zprávu, která má být vložena do fronty nebo získána z fronty. Obsahuje vyrovnávací paměť a zapouzdřuje data aplikace i 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 voleb MQGetMessagepředstavuje strukturu MQGMO. Má vlastnosti odpovídající polím MQGMO.

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

Multi **MQDistributionList**
Objekt třídy MQDistributionList představuje distribuční seznam (používaný k odesílání více zpráv s jedním MQPUT). Obsahuje seznam objektů položek MQDistributionList.

Multi **Položka MQDistributionList**
Objekt třídy položek MQDistributionListpředstavuje jeden cíl 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, produkt IBM MQ vrací manipulátory připojení a manipulátory objektů do programu.

Tyto popisovače musí být předány jako parametry při následných voláních IBM MQ . Pomocí objektového modelu IBM MQ jsou tyto popisovače v aplikačním programu skryty. Místo toho vytvoření objektu ze třídy vede k vrácení odkazu na objekt do aplikačního programu. Jedná se o tento odkaz na objekt, který se používá při volání metod a přístupu k vlastnostem vůči objektu.

Návratové kódy

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

Tyto návratové kódy jsou kódem dokončení a kódem příčiny a jsou samy o sobě vlastnostmi objektu. Hodnoty kódu dokončení a kódu příčiny jsou stejné jako ty, které jsou definovány pro rozhraní MQI, s některými dalšími hodnotami specifickými pro objektově orientované prostředí.

Zprávy produktu IBM MQ

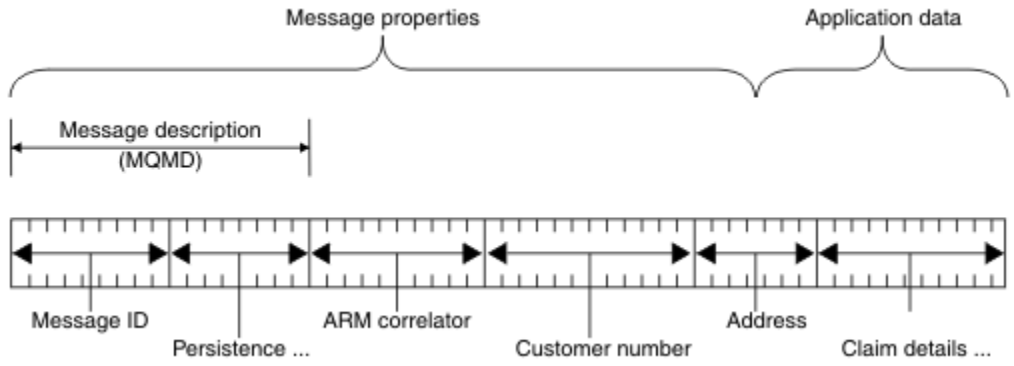
Zpráva IBM MQ se skládá z vlastností zprávy a dat aplikace. Deskriptor zpráv fronty zpráv (MQMD) obsahuje řídicí informace, které doprovázejí data aplikace, když zpráva prochází mezi odesílající a přijímající aplikací.

Části zprávy

Zprávy IBM MQ se skládají ze dvou částí:

- Vlastnosti zprávy
- Data aplikací

Obrázek 2 na stránce 18 představuje zprávu a ukazuje, jak je logicky rozdělena na vlastnosti zprávy a data aplikace.



Obrázek 2. Reprezentace zprávy

Data aplikace, která jsou přenášena ve zprávě IBM MQ , nejsou správcem front změněna, pokud není na ní proveden převod dat. IBM MQ také neuvádí žádná omezení obsahu těchto dat. Délka dat v každé zprávě nesmí překročit hodnotu atributu **MaxMsgLength** fronty i správce front.

ALW V systému AIX, Linux, and Windowsje výchozí hodnota atributu *MaxMsgLength* správce front a fronty 4 MB (4 194 304 bajtů), kterou lze v případě potřeby změnit na maximálně 100 MB (104 857 600 bajtů).

IBM i V systému IBM ije výchozí hodnota atributu *MaxMsgLength* správce front a fronty 4 MB (4 194 304 bajtů), kterou lze v případě potřeby změnit na maximálně 100 MB (104 857 600 bajtů). Hodláte-li v systému IBM ipoužívat zprávy systému IBM MQ větší než 15 MB, přečtěte si téma “Sestavení vaší procedurální aplikace na IBM i” na stránce 973.

z/OS V systému z/OSje atribut **MaxMsgLength** správce front opraven na hodnotu 100 MB a atribut **MaxMsgLength** fronty má výchozí hodnotu 4 MB (4 194 304 bajtů), kterou lze v případě potřeby změnit na maximálně 100 MB.

Za určitých okolností by vaše zprávy byly o něco kratší než hodnota atributu **MaxMsgLength** . Další informace viz téma “Data ve vaší zprávě” na stránce 734.

Zprávu vytvoříte při použití volání MQPUT nebo MQPUT1 MQI. Jako vstup pro tato volání zadáte řídicí informace (například prioritu zprávy a název 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 viz MQMD-deskriptor zprávy.

Popis použití polí v deskriptoru MQMD, která obsahují informace o původu zprávy, naleznete v tématu “kontext zprávy” na stránce 45.

Existují různé verze deskriptoru zprávy. Další informace o seskupování a segmentaci zpráv (viz “Skupiny zpráv” na stránce 42) jsou uvedeny ve verzi 2 deskriptoru zpráv (nebo MQMDE). Toto je stejné jako deskriptor zprávy verze 1, ale obsahuje další pole. Tato pole jsou popsána v části MQMDE-Rozšíření deskriptoru zpráv.

Typy zpráv

Existují čtyři typy zpráv definované pomocí IBM MQ.

Tyto čtyři zprávy jsou:

- Datagram
- Vyžádat zprávy
- Zprávy odpovědí
- Zprávy sestavy
  - Typy zpráv sestavy
  - Volby zprávy sestavy

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

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

Datagramy

Použijte *datagram* , když nepožadujete odpověď od aplikace, která přijme zprávu (tj. získá zprávu z fronty).

Příkladem aplikace, která může používat datagramy, je aplikace, která zobrazuje informace o letu v letištním salónku. Zpráva může obsahovat data pro celou obrazovku letových informací. Je nepravděpodobné, že by taková aplikace požadovala potvrzení pro zprávu, protože pravděpodobně nezáleží na tom, zda zpráva nebyla doručena. Aplikace po krátké době odešle zprávu o aktualizaci.

Zprávy požadavku

Použijte *zprávu požadavku* , když chcete odpověď od aplikace, která přijme zprávu.

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

Chcete-li propojit zprávu odpovědi se zprávou požadavku, máte dvě možnosti:

- Zajistěte, aby aplikace, která zpracovává zprávu požadavku, byla zodpovědná za to, že vloží informace do zprávy odpovědi, která souvisí se zprávou požadavku.
- Pole sestavy v deskriptoru zprávy požadavku použijte k určení obsahu polí *MsgId* a *CorrelId* zprávy odpovědi:
  - Můžete požadovat, aby se buď *MsgId* , nebo *CorrelId* původní zprávy zkopírovaly do pole *CorrelId* zprávy odpovědi (výchozí akce je kopírovat *MsgId*).

– Můžete požádat o vygenerování nového souboru *MsgId* pro zprávu odpovědi nebo o zkopírování položky *MsgId* původní zprávy do pole *MsgId* zprávy odpovědi (výchozí akcí je vygenerování nového identifikátoru zprávy).

Zprávy odpovědi

Použijte *zprávu odpovědi* , když odpovíte na jinou zprávu.

Při vytváření zprávy odpovědi respektujte všechny volby, které byly nastaveny v deskriptoru zprávy, na kterou odpovídáte. Volby sestavy určují obsah polí identifikátoru zprávy (*MsgId*) a identifikátoru korelace (*CorrelId*). Tato pole umožňují aplikaci, která obdrží odpověď, korelovat odpověď s původním požadavkem.

Zprávy sestavy

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

Mohou být generovány:

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

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

Typy zpráv sestavy

Když vložíte zprávu do fronty, můžete vybrat příjem:

- *Zpráva sestavy výjimek*. Tato zpráva je odeslána jako odpověď na zprávu se nastaveným příznakem výjimek. Je generován agentem kanálu zpráv (MCA) nebo aplikací.
- *Zpráva sestavy vypršení platnosti*. To znamená, že se aplikace pokusila načíst zprávu, která dosáhla prahové hodnoty vypršení platnosti; zpráva je označena jako vyřazená. Tento typ sestavy je generován správcem front.
- *Zpráva sestavy potvrzení o přijetí (COA)*. To označuje, že zpráva dosáhla své cílové fronty. Je generován správcem front.
- *Zpráva sestavy potvrzení o doručení (COD)*. To označuje, že zpráva byla načtena přijímající aplikací. Je generován správcem front.
- *Zpráva sestavy oznámení pozitivní akce (PAN)*. To označuje, že požadavek byl úspěšně obsloužen (to znamená, že akce požadovaná ve zprávě byla úspěšně provedena). Tento typ sestavy je generován aplikací.
- *Zpráva sestavy oznámení negativní akce (NAN)*. To označuje, 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 a volby zprávy se zprávou o výjimce, obdržíte zprávu se zprávou o výjimce. Pokud však vyberete pouze volbu zprávy se zprávou o potvrzení doručení a zpráva se nedoručí, neobdržíte zprávu se zprávou o výjimce.

Zprávy sestavy, které požadujete, když jsou splněna kritéria pro generování konkrétní zprávy, jsou jediné, které obdržíte.

Volby zprávy sestavy

Po výskytu výjimky můžete zprávu *vyřadit* . Pokud vyberete volbu vyřazení a požadujete zprávu sestavy výjimek, zpráva sestavy přejde na *ReplyToQ* a *ReplyToQMgr*a původní zpráva bude vyřazena.

Poznámka: Výhodou je, že můžete snížit počet zpráv směřujících do fronty nedoručených zpráv. To však znamená, že vaše aplikace, pokud neodešle pouze datagramové zprávy, se musí vypořádat s vrácenými zprávami. Když je vygenerována zpráva sestavy výjimek, zdědí perzistenci původní zprávy.

Pokud zprávu sestavy nelze doručit (například pokud je fronta plná), zpráva sestavy se umístí do fronty nedoručených zpráv.

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

Pomocí dalších voleb sestavy v deskriptoru zprávy (MQMD) můžete určit obsah polí *MsgId* a *CorrelId* všech zpráv sestavy vytvořených pro zprávu:

- Můžete požadovat, aby se buď *MsgId* , nebo *CorrelId* původní zprávy zkopírovaly do pole *CorrelId* zprávy sestavy. Výchozí akce je kopírovat identifikátor zprávy. Použijte MQRO_COPY_MSG_ID_TO_CORRELID, protože umožňuje odesílateli zprávy korelovat odpověď nebo zprávu sestavy s původní zprávou. Identifikátor korelace odpovědi nebo zprávy sestavy je identický s identifikátorem zprávy původní zprávy.
- Můžete požadovat, aby byl pro zprávu sestavy vygenerován nový soubor *MsgId* , nebo aby byl soubor *MsgId* původní zprávy zkopírován do pole *MsgId* zprávy sestavy. Výchozí akce je generovat nový identifikátor zprávy. Použijte MQRO_NEW_MSG_ID, protože zajišťuje, že každá zpráva v systému má jiný identifikátor zprávy a lze ji jednoznačně rozlišit od všech ostatních zpráv v systému.
- Specializované aplikace mohou potřebovat použít MQRO_PASS_MSG_ID nebo MQRO_PASS_CORREL_ID. Musíte však navrhnout aplikaci, která čte zprávy z fronty, abyste se ujistili, že správně funguje, například když fronta obsahuje více zpráv se stejným identifikátorem zprávy.

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

  Aplikace, které se chovají jako prostředníci mezi aplikací žadatele a serverovou aplikací, nemusí kontrolovat nastavení těchto příznaků. Důvodem je skutečnost, že tyto aplikace obvykle potřebují předat zprávu serverové aplikaci s poli *MsgId*, *CorrelId*a *Report* beze změny. To umožňuje serverové aplikaci kopírovat soubor *MsgId* z původní zprávy v poli *CorrelId* zprávy odpovědi.

Při generování sestavy o zprávě musí serverové aplikace otestovat, zda byla některá z těchto voleb nastavena.

Další informace o tom, jak používat zprávy sestavy, viz Sestava.

K označení povahy sestavy používají správci front rozsah kódů zpětné vazby. Tyto kódy vloží do pole *Feedback* deskriptoru zprávy zprávy sestavy. Správci front mohou také vrátit kódy příčiny MQI v poli *Feedback* . IBM MQ definuje rozsah kódů zpětné vazby pro použití aplikacemi.

Další informace o zpětné vazbě a kódech příčin naleznete v tématu Zpětná vazba.

Příkladem programu, který by mohl používat kód zpětné vazby, je program, který monitoruje pracovní zátěž ostatních programů obsluhujících frontu. Pokud existuje více než jedna instance programu, který obsluhuje frontu, a počet zpráv, které do fronty dorazí, to již neodůvodňuje, může takový program odeslat zprávu sestavy (s kódem zpětné vazby MQFB_QUIT) jednomu z obsluhujících programů, aby označil, že by program měl ukončit svou činnost. (Program monitorování může pomocí volání MQINQ zjistit, kolik programů obsluhuje frontu.)

Multi *Sestavy a segmentované zprávy*

Není podporováno na systému IBM MQ for z/OS.

Pokud je zpráva segmentována a požádáte o vygenerování sestav, můžete obdržet více sestav, než byste dělali, kdyby zpráva nebyla segmentována.

Popis segmentovaných zpráv viz “Segmentace zpráv” na stránce 767.

Pro sestavy generované produktem IBM MQ

Pokud segmentujete zprávy nebo povolíte to správci front, existuje pouze jeden případ, kdy můžete očekávat, že obdržíte jedinou sestavu pro celou zprávu. Toto je, když jste požadovali pouze sestavy COD a zadali jste MQGMO_COMPLETE_MSG v aplikaci získávání.

V ostatních případech musí být vaše žádost připravena na řešení několika sestav; obvykle jedna pro každý segment.

Poznámka: Pokud segmentujete zprávy a potřebujete vrátit pouze prvních 100 bajtů původních dat zprávy, změňte nastavení voleb sestavy tak, abyste požádali o sestavy bez dat pro segmenty, které mají offset 100 nebo více. Pokud tak neučiníte a ponecháte nastavení tak, aby každý segment požadoval 100 bajtů dat, a načtete zprávy sestavy s jedním MQGET určujícím MQGMO_COMPLETE_MSG, sestavy se sestaví do velké zprávy obsahující 100 bajtů přečtených dat při každém odpovídajícím posunu. Pokud k tomu dojde, potřebujete velkou vyrovnávací paměť nebo musíte zadat MQGMO_ACCEPT_TRUNCATED_MSG.

Pro sestavy generované aplikacemi

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

Pak přidejte žádná, 100 bajtů nebo všechna původní data zprávy (nebo jakékoli jiné množství, které byste obvykle zahrnuli) k datům zprávy sestavy.

Můžete rozpoznat záhlaví IBM MQ , která musí být zkopírována, pomocí následujících názvů formátu, počínaje MQMD a pokračováním v libovolných přítomných záhlavích. Následující názvy `Format` označují tato záhlaví IBM MQ :

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

MQH* znamená 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 také vyskytuje na stejné pozici pro MQMDE, MQIMSa všechna záhlaví MQH*.

Používáte-li verzi 1 MQMD a vykazujete-li segment nebo zprávu ve skupině nebo zprávu, pro kterou je segmentace povolena, musí data sestavy začínat na MQMDE. Nastavte pole *`OriginalLength`* na délku původních dat zprávy s vyloučením délek všech záhlaví IBM MQ , která najdete.

Načítání sestav

Pokud požádáte o sestavy COA nebo COD, můžete požádat o jejich opětovné sestavení pomocí MQGMO_COMPLETE_MSG.

Příkaz MQGET s MQGMO_COMPLETE_MSG je splněn, když je ve frontě dostatek zpráv sestavy (jednoho typu, například COA, a se stejným *`GroupId`*), které představují jednu úplnou původní zprávu. To platí i v případě, že samotné zprávy sestavy neobsahují úplná původní data; pole *`OriginalLength`* v každé zprávě sestavy uvádí délku původních dat představovaných touto zprávou sestavy, i když samotná data nejsou přítomna.

Tuto techniku můžete použít i v případě, že fronta obsahuje několik různých typů sestav (například COA i COD), protože příkaz MQGET s MQGMO_COMPLETE_MSG znovu sestavuje zprávy sestavy pouze v případě, že mají stejný kód *Feedback* . Tuto techniku však obvykle nemůžete použít pro sestavy výjimek, protože obecně mají různé kódy *Feedback* .

Tuto techniku můžete použít k získání pozitivního označení, že dorazila celá zpráva. Ve většině případů však musíte zajistit možnost, že některé segmenty dorazí, zatímco jiné mohou generovat výjimku (nebo vypršení platnosti, pokud jste to povolili). V tomto případě nemůžete použít MQGMO_COMPLETE_MSG, protože obecně můžete získat různé kódy *Feedback* pro různé segmenty a můžete získat více než jednu sestavu pro segment. Můžete však použít MQGMO_ALL_SEGMENTS_AVAILABLE.

Chcete-li to povolit, možná budete muset načíst sestavy při jejich doručení a vytvořit obrázek v aplikaci, co se stalo s původní zprávou. Pomocí pole *GroupId* ve zprávě sestavy můžete korelovat sestavy s *GroupId* původní zprávy a pomocí pole *Feedback* můžete identifikovat typ každé zprávy sestavy. Způsob, jakým to provedete, závisí na požadavcích vaší aplikace.

Jedním z přístupů je následující:

- Požádejte o sestavy COD a sestavy výjimek.
- Po specifickém čase zkontrolujte, zda byla přijata úplná sada sestav COD pomocí MQGMO_COMPLETE_MSG. Pokud ano, vaše aplikace ví, že celá zpráva byla zpracována.
- Pokud ne, a sestavy výjimek vztahující se k této zprávě jsou přítomny, vyřešte problém jako u nesegmentovaných zpráv, ale ujistěte se, že v určitém okamžiku vyčistíte osiřelé segmenty.
- Pokud existují segmenty, pro které neexistují žádné sestavy jakéhokoli druhu, mohou původní segmenty (nebo sestavy) čekat na opětovné připojení kanálu, nebo může být síť v určitém bodě přetížena. Pokud nebyly obdrženy žádné zprávy o výjimkách (nebo pokud se domníváte, že ty, které máte, mohou být pouze dočasné), můžete se rozhodnout nechat vaši aplikaci déle čekat.

  Stejně jako dříve je to podobné aspektům, které máte při práci s nesegmentovanými zprávami, kromě toho, že musíte také zvážit možnost vyčištění osiřelých segmentů.

Pokud původní zpráva není kritická (například pokud se jedná o dotaz nebo zprávu, kterou lze později opakovat), nastavte dobu vypršení platnosti, abyste zajistili odebrání osiřelých segmentů.

Správci front nižší úrovně

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

Pokud však segment zprávy prochází správcem front, který nepodporuje segmentaci, a pokud je zde vygenerována sestava, bude struktura MQMDE v původní zprávě považována za čistě datovou. Proto není zahrnut do dat sestavy, pokud bylo požadováno nula bajtů původních dat. Bez MQMDE nemusí být zpráva sestavy užitečná.

Požadujte alespoň 100 bajtů dat v sestavách, pokud existuje možnost, že by zpráva mohla procházet správcem front nižší úrovně.

Formát řídicích informací zpráv a dat zpráv

Správce front má zájem pouze o formát řídicích informací v rámci zprávy, zatímco aplikace, které zpracovávají zprávu, mají zájem o formát řídicích informací i dat.

Formát informací o řízení zpráv

Řídicí informace v polích znakového řetězce deskriptoru zprávy 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 přenášejí zprávy, používají hodnotu tohoto atributu k určení, jaký převod dat se má provést.

Formát dat zprávy

Můžete zadat libovolnou z následujících věcí:

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

Chcete-li to provést, použijte tato pole:

Format
To označuje příjemci 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 zprávu do fronty nedoručených zpráv (nedoručených zpráv). Přidá do zprávy záhlaví (obsahující více řídicích informací) a změní pole *Format* tak, aby toto zobrazilo.

Správce front má řadu *vestavěných formátů* s názvy začínajícími na MQ, například MQFMT_STRING. Pokud tyto formáty nesplňují vaše potřeby, můžete pro ně definovat vlastní formáty (*formáty definované uživatelem*), ale nesmíte pro ně používat názvy začínající řetězcem MQ .

Při vytváření a používání vlastních formátů je třeba napsat uživatelskou proceduru pro převod dat, která bude podporovat program, který zprávu získává pomocí příkazu MQGMO_CONVERT.

CodedCharSetId
Definuje znakovou sadu znakových dat ve zprávě. Chcete-li tuto znakovou sadu nastavit na znakovou sadu správce front, můžete toto pole nastavit na konstantní hodnotu 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 převést libovolná 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, pakovaná desetinná celá čísla a čísla s pohyblivou řádovou čárkou. Obvykle je zakódován podle konkrétního počítače, na kterém je spuštěn správce front.

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

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

Převod dat aplikace

Data aplikace mohou vyžadovat převod na znakovou sadu a kódování požadované jinou aplikací, pokud se jedná o různé platformy.

Lze jej převést v odesílajícím správci front nebo v přijímajícím správci front. Pokud knihovna vestavěných formátů nesplňuje vaše potřeby, můžete definovat vlastní. Typ převodu závisí na formátu zprávy, který je určen v poli formátu deskriptoru zprávy MQMD.

Poznámka: Zprávy se zadaným MQFMT_NONE nejsou převedeny.

Převod v odesílajícím správci front

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

Převod se provádí v odesílajícím správci front pro určité vestavěné formáty a pro formáty definované uživatelem, pokud je zadána vhodná uživatelská procedura.

Vestavěné formáty

Patří k nim:

- Zprávy, které jsou všechny znaky (pomocí názvu formátu MQFMT_STRING)
- IBM MQ definované zprávy, například programovatelné formáty příkazů

  Produkt IBM MQ používá zprávy formátu programovatelných příkazů pro administrativní zprávy a události (v tomto případě se používá název formátu MQFMT_ADMIN). Pro vlastní zprávy můžete použít stejný formát (pomocí názvu formátu MQFMT_PCF) a využít vestavěný převod dat.

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

Formáty definované aplikací

V případě formátů definovaných uživatelem musí být konverze dat aplikace provedena uživatelským programem pro převod dat (další informace viz “Zápis uživatelských procedur převodu dat” na stránce 950). V prostředí klient-server je uživatelská procedura načtena na serveru a probíhá zde převod.

Převod v přijímajícím správci front

Přijímající správce front může převést data zpráv aplikace pro vestavěné i uživatelem definované formáty.

Převod se provádí během zpracování volání MQGET, pokud zadáte volbu MQGMO_CONVERT. Podrobnosti viz Volby .

Kódované znakové sady

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

Při vytváření správce front je použité ID kódované znakové sady (CCSID) správce front založeno na ID základního prostředí. Pokud se jedná o smíšenou kódovou stránku, produkt IBM MQ použije jako CCSID správce front část SBCS smíšené kódové stránky.

Pro obecnou konverzi dat platí, že pokud základní operační systém podporuje kódové stránky DBCS, může je použít IBM MQ .

Podrobnosti o kódovaných znakových sadách, které podporuje, naleznete v dokumentaci k vašemu operačnímu systému.

Při psaní aplikací, které pokrývají více platforem, je třeba vzít v úvahu převod dat aplikace, názvy formátů a uživatelské procedury. Informace o vyvolání a zápisu uživatelských procedur pro převod dat viz “Zápis uživatelských procedur převodu dat” na stránce 950 .

Priority zpráv

Můžete buď nastavit prioritu zprávy na číselnou hodnotu, nebo nechat zprávu převzít výchozí prioritu fronty.

Prioritu zprávy (v poli *Priority* struktury MQMD) nastavíte při jejím vložení do fronty. Můžete nastavit číselnou hodnotu priority nebo nechat zprávu, aby přijala výchozí prioritu fronty.

Atribut **MsgDeliverySequence** fronty určuje, zda jsou zprávy ve frontě uloženy v pořadí FIFO (první dovnitř, první ven) nebo v pořadí FIFO v rámci priority. Je-li tento atribut nastaven na hodnotu MQMDS_PRIORITY, jsou zprávy zařazeny do fronty s prioritou určenou v poli *Priority* jejich deskriptorů

zpráv; je-li však nastavena na hodnotu MQMDS_FIFO, jsou zprávy zařazeny do fronty s výchozí prioritou fronty. Zprávy se stejnou prioritou jsou uloženy 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 fronty a lokální definice vzdálených front mohou mít jiné výchozí priority než základní fronty, na které se vyřeší. Pokud je v cestě rozlišení více než jedna definice fronty (viz "Rozlišení názvu" na stránce 722), výchozí priorita je převzata z hodnoty (v době operace vložení) atributu **DefPriority** fronty uvedené v příkazu open.

Hodnotou atributu **MaxPriority** správce front je maximální priorita, kterou lze přiřadit ke zprávě zpracované daným správcem front. Hodnotu tohoto atributu nemůžete změnit. V produktu IBM MQmá atribut hodnotu 9; můžete vytvořit zprávy s prioritami mezi 0 (nejnižší) a 9 (nejvyšší).

Vlastnosti zprávy

Pomocí vlastností zprávy můžete aplikaci povolit výběr zpráv ke zpracování nebo načtení informací o zprávě bez přístupu k záhlavím MQMD nebo MQRFH2 . Také usnadňují komunikaci mezi aplikacemi IBM MQ a JMS .

Vlastnost zprávy jsou data přidružená ke zprávě, která se skládají z textového názvu a hodnoty konkrétního typu. Vlastnosti zpráv používají selektory zpráv k filtrování publikování na témata nebo k selektivnímu získání zpráv z front. Vlastnosti zprávy lze použít k zahrnutí obchodních dat nebo informací o stavu, aniž by bylo nutné je ukládat do dat aplikace. Aplikace nemusí přistupovat k datům v záhlaví MQMD (MQ Message Descriptor) nebo MQRFH2 , protože k polím v těchto datových strukturách lze přistupovat jako k vlastnostem zpráv pomocí volání funkce MQI (Message Queue Interface).

Použití vlastností zprávy v souboru IBM MQ napodobuje použití vlastností v souboru JMS. To znamená, že můžete nastavit vlastnosti v aplikaci JMS a načíst je v procedurální aplikaci IBM MQ nebo naopak. Chcete-li vlastnost zpřístupnit aplikaci JMS , přiřaďte jí předponu "usr"; pak je k dispozici (bez předpony) jako vlastnost uživatele zprávy JMS . Například vlastnost IBM MQ property *usr.myproperty* (řetězec znaků) je přístupná pro aplikaci JMS pomocí volání JMS `message.getStringProperty('myproperty')`. Všimněte si, že aplikace JMS nemohou přistupovat k vlastnostem s předponou "usr", pokud obsahují dva nebo více U+002E (".") znaků. Vlastnost bez předpony a bez předpony U+002E (".") S znakem se zachází, jako by měl předponu "usr". Naopak k uživatelské vlastnosti nastavené v aplikaci JMS lze přistupovat v aplikaci IBM MQ přidáním řetězce "usr." Předpona názvu vlastnosti, na kterou se dotazuje volání MQINQMP.

Vlastnosti zprávy a délka zprávy

Pomocí atributu správce front *MaxPropertiesDélka* můžete řídit velikost vlastností, které mohou proudit s libovolnou zprávou ve správci front IBM MQ .

Obecně platí, že použijete-li k nastavení vlastností MQSETMP, velikost vlastnosti je délka názvu vlastnosti v bajtech plus délka hodnoty vlastnosti v bajtech předané do volání MQSETMP. Je možné, aby se znaková sada názvu vlastnosti a hodnoty vlastnosti změnila během přenosu zprávy do jejího místa určení, protože je lze převést na Unicode; v tomto případě se může změnit velikost vlastnosti.

Ve volání MQPUT nebo MQPUT1 se vlastnosti zprávy nepočítají do délky zprávy pro frontu a správce front, ale počítají se do délky vlastností, jak jsou vnímány správcem front (bez ohledu na to, zda byly nastaveny pomocí volání MQI vlastnosti zprávy).

Pokud velikost vlastností překročí maximální délku vlastností, bude zpráva odmítnuta s hodnotou MQRC_PROPERTIES_TOO_BIG. Protože velikost vlastností závisí na jejich reprezentaci, měli byste nastavit maximální délku vlastností na hrubé úrovni.

Je možné, aby aplikace úspěšně vložila zprávu s vyrovnávací pamětí, která je větší než hodnota *MaxMsgLength*, pokud vyrovnávací paměť obsahuje vlastnosti. Důvodem je, že i když jsou reprezentovány prvky MQRFH2 , nepočítají se vlastnosti zprávy do délky 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 je v záhlaví MQRFH2 obsažena jedna nebo více složek a libovolná složka neobsahuje vlastnosti, pole záhlaví MQRFH2 se místo toho započítávají do délky zprávy.

Ve volání MQGET se vlastnosti zprávy nepočítají do délky zprávy, pokud jde o frontu a správce front. Protože jsou však vlastnosti počítány odděleně, je možné, že vyrovnávací paměť vrácená voláním MQGET je větší než hodnota atributu *MaxMsgLength* .

Nedotazujte aplikace na hodnotu *MaxMsgLength* a poté přidělte vyrovnávací paměť této velikosti před voláním MQGET; místo toho přidělte vyrovnávací paměť, kterou považujete za dostatečně velkou. Pokud se příkaz MQGET nezdaří, přidělte vyrovnávací paměť řízenou velikostí parametru *DataLength* .

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

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

Existuje limit délky 100 MB pro vlastnosti zprávy, s výjimkou deskriptoru zprávy nebo rozšíření pro každou zprávu.

Velikost vlastnosti v její interní reprezentaci je délka názvu plus velikost její hodnoty plus některá řídicí data pro vlastnost. Po přidání jedné vlastnosti do zprávy existují také některá řídicí data pro sadu vlastností.

Názvy vlastností

Název vlastnosti je znakový řetězec. Na délku a sadu znaků, které lze použít, se vztahují určitá omezení.

Název vlastnosti je znakový řetězec rozlišující velikost písmen, omezený na +4095 znaků, pokud kontext nestanoví jinak. Toto omezení je obsaženo v konstantě MQ_MAX_PROPERTY_NAME_LENGTH.

Pokud tuto maximální délku překročíte při použití volání MQI vlastnosti zprávy, volání se nezdaří s kódem příčiny MQRC_PROPERTY_NAME_LENGTH_ERR.

Protože v souboru JMSnení maximální délka názvu vlastnosti, je možné, aby aplikace JMS nastavila platný název vlastnosti JMS , který není platným názvem vlastnosti IBM MQ , když je uložen ve struktuře MQRFH2 .

V tomto případě se při analýze použije pouze prvních 4095 znaků názvu vlastnosti; následující znaky se ořízno u. To může způsobit, že se aplikaci používající selektory nepodaří najít shodu s řetězcem výběru, nebo že bude odpovídat řetězci, pokud to neočekává, protože více než jedna vlastnost může být zkrácena na stejný název. Při oříznutí názvu vlastnosti produkt WebSphereMQ vydá zprávu protokolu chyb.

Všechny názvy vlastností musí odpovídat pravidlům definovaným ve specifikaci jazyka Java pro identifikátory Java s tou výjimkou, že znak Unicode U+002E (.) je povolen jako součást názvu-nikoli však jako začátek. Pravidla pro identifikátory Java odpovídají těm, které jsou obsaženy ve specifikaci JMS pro názvy vlastností.

Znaky mezer a operátory porovnání jsou zakázány. Vložené hodnoty null jsou povoleny v názvu vlastnosti, ale nejsou doporučeny. Používáte-li vložené hodnoty Null, zabráníte použití konstanty MQVS_NULL_TERMINATED při použití se strukturou MQCHARV k určení řetězců s proměnnou délkou.

Zachovejte názvy vlastností jednoduché, protože aplikace mohou vybírat zprávy na základě názvů vlastností a převod mezi znakovou sadou názvu a selektorem může způsobit neočekávané selhání výběru.

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

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

- Body
- Properties

Dobrým způsobem, jak se vyhnout kolizím názvů, je zajistit, aby všechny aplikace předřadily své vlastnosti zpráv svým názvem internetové domény. Pokud například vyvíjíte aplikaci s názvem domény ourcompany.com , můžete všechny vlastnosti pojmenovat předponou com.ourcompany. Tato konvence pojmenování také umožňuje snadný výběr vlastností; například aplikace se může dotázat na všechny vlastnosti zprávy začínající na com.ourcompany.%.

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

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

Pro názvy vlastností platí následující omezení:

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

   Jedinými výjimkami jsou následující vlastnosti poskytující synonyma pro vlastnosti JMS :

Vlastnost	**Synonymum pro**
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
JMSRedelivered	Kořen.MQMD.BackoutCount
JMSReplyTo (řetězec kódovaný jako identifikátor URI)	Kořen.MQMD.ReplyToQ nebo Root.MQMD.ReplyToQMgr nebo jms.Rto
JMSTimestamp	Kořen.MQMD.PutDate nebo Root.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 aplikaci MQI přistupovat k vlastnostem JMS podobným způsobem jako klientská aplikace IBM MQ classes for JMS . Z těchto vlastností lze pomocí rozhraní MQI nastavit pouze JMSCorrelationID, JMSReplyTo, JMSType, JMSXGroupIDa JMSXGroupSeq .

   Všimněte si, že vlastnosti JMS_IBM_ * dostupné v rámci produktu IBM MQ classes for JMS nejsou k dispozici pomocí rozhraní MQI. K polím, na která odkazují vlastnosti JMS_IBM_ *, lze přistupovat jinými způsoby pomocí aplikací MQI.
2. Vlastnost nesmí být volána v žádné kombinaci malých nebo velkých písmen, "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 začíná " mq " v jakékoli směsi malých nebo velkých písmen a nezačínající "mq_usr" může obsahovat pouze jednu "." znak (U+002E). Více "." ve vlastnostech s těmito předponami nejsou povoleny znaky.
4. Dvě "." Znaky musí mezi sebou obsahovat jiné znaky; v hierarchii nemůžete mít prázdný bod. Podobně název vlastnosti nemůže končit na "." Znak.
5. Pokud aplikace nastaví vlastnost "a.b" a poté vlastnost "a.b.c", není jasné, zda v hierarchii "b" obsahuje hodnotu nebo jiné logické seskupení. Taková hierarchie je "smíšený obsah" a není podporována. Nastavení vlastnosti, která způsobuje smíšený obsah, není povoleno.

Tato omezení jsou vynucována mechanizmem ověřování následujícím způsobem:

- Názvy vlastností se ověřují při nastavování vlastnosti pomocí volání MQSETMP- Nastavte vlastnost zprávy , pokud bylo při vytváření popisovače zprávy požadováno ověření. Pokud je proveden pokus o ověření vlastnosti a nezdaří se kvůli chybě ve specifikaci názvu vlastnosti, kód dokončení je MQCC_FAILED s příčinou:
  - MQRC_PROPERTY_NAME_ERROR z důvodů 1-4.
  - MQRC_MIXED_CONTENT_NOT_ALLOWED pro příčinu 5.
- Není zaručeno, že názvy vlastností zadané přímo jako prvky MQRFH2 budou ověřeny voláním MQPUT.

Pole deskriptoru zpráv jako vlastnosti
S většinou polí deskriptoru zpráv lze zacházet jako s vlastnostmi. Název vlastnosti je vytvořen přidáním předpony k názvu pole deskriptoru zprávy.

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

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

Zadejte hodnotu *Field* se stejnou velikostí písmen jako pro pole struktury MQMD v deklaraci jazyka C. Například název vlastnosti `Root.MQMD.AccountingToken` přistupuje k poli `AccountingToken` deskriptoru zprávy.

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

Pole deskriptoru zpráv nejsou nikdy reprezentována v záhlaví MQRFH2 jako u jiných vlastností.

Pokud data zprávy začínají hodnotou MQMDE, kterou správce front uznává, lze k polím MQMDE přistupovat pomocí popsané notace `Root.MQMD.`*`Field`* . V tomto případě jsou pole MQMDE považována z perspektivy vlastností za logicky součást MQMD. Viz téma Přehled MQMDE.

Datové typy a hodnoty vlastností

Vlastnost může být logická hodnota, bajtový řetězec, znakový řetězec nebo číslo s pohyblivou řádovou čárkou nebo celé číslo. Vlastnost může ukládat libovolnou platnou hodnotu v rozsahu datového typu, pokud to kontext dále neomezuje.

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; je to vlastnost s hodnotou null. Vlastnost s hodnotou null se liší od bajtové vlastnosti (MQBYTE []) nebo vlastnosti znakového řetězce (MQCHAR []) v tom, že má definovanou, ale prázdnou hodnotu, tj. hodnotu s nulovou délkou.

Bajtový řetězec není platný datový typ vlastnosti v JMS nebo XMS. Doporučuje se nepoužívat vlastnosti bajtového řetězce ve složce *usr* .

Výběr zpráv z front

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

Selektory.

Selektor zpráv je řetězec proměnné délky používaný aplikací k registraci jejího zájmu pouze o ty zprávy, které mají vlastnosti splňující dotaz SQL (Structured Query Language), který představuje výběrový řetězec.

Výběr pomocí volání funkcí MQSUB a MQOPEN

K provedení výběru pomocí volání MQSUB a MQOPEN použijte *SelectionString*, což je struktura typu MQCHARV.

Struktura *SelectionString* se používá k předání řetězce výběru proměnné délky správci front.

CCSID přidružený k řetězci selektoru je nastaven prostřednictvím pole VSCCSID struktury MQCHARV. Použitá hodnota musí být CCSID, který je podporován pro řetězce selektoru. Seznam podporovaných kódových stránek viz Převod kódových stránek .

Určení CCSID, pro který neexistuje žádný IBM MQ podporovaný převod Unicode, má za následek chybu MQRC_SOURCE_CCSID_ERROR. Tato chyba je vrácena v okamžiku, kdy je selektor prezentován správci front, tj. ve volání MQSUB, MQOPEN nebo MQPUT1 .

Výchozí hodnota pro pole *VSCCSID* je MQCCSI_APPL, která označuje, že CCSID řetězce výběru je stejné jako CCSID správce front, nebo CCSID klienta, pokud je připojen prostřednictvím klienta. Konstantu MQCCSI_APPL však může před kompilací přepsat aplikace, která ji předefinovala.

Pokud selektor MQCHARV představuje řetězec s hodnotou NULL, nebude pro daného spotřebitele zpráv proveden žádný výběr a zprávy budou doručeny, jako by nebyl selektor použit.

Maximální délka řetězce výběru je omezena pouze tím, co lze popsat v poli MQCHARV *VSLength*.

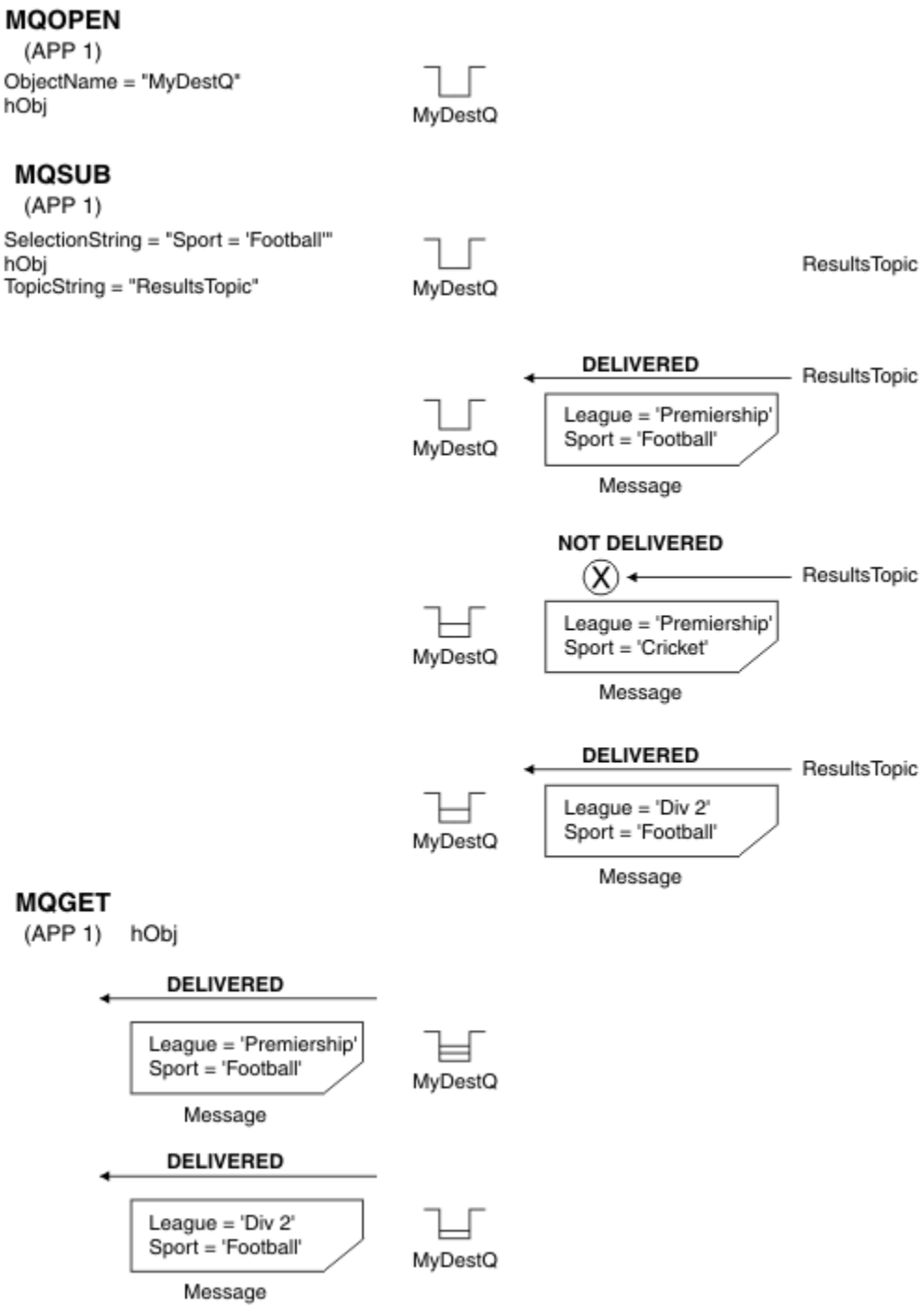
Hodnota SelectionString je vrácena na výstupu volání MQSUB s použitím volby odběru MQSO_RESUME, pokud jste zadali vyrovnávací paměť a v parametru VSBufSizeje kladná délka vyrovnávací paměti. Pokud vyrovnávací paměť nezadáte, bude v poli VSLlength v tabulce MQCHARV vrácena pouze délka řetězce výběru. Je-li poskytnutá vyrovnávací paměť menší než prostor požadovaný pro vrácení pole, jsou v poskytnuté vyrovnávací paměti vráceny pouze bajty VSBufSize .

Aplikace nemůže změnit řetězec výběru bez předchozího zavření manipulátoru fronty (pro MQOPEN) nebo odběru (pro MQSUB). V následném volání MQOPEN nebo MQSUB lze poté zadat nový řetězec výběru.

MQOPEN
Pomocí příkazu MQCLOSE zavřete otevřený manipulátor a poté zadejte nový řetězec výběru pro následné volání MQOPEN.

MQSUB
Pomocí příkazu MQCLOSE zavřete vrácený manipulátor odběru (hSub) a poté zadejte nový řetězec výběru pro následné volání MQSUB.

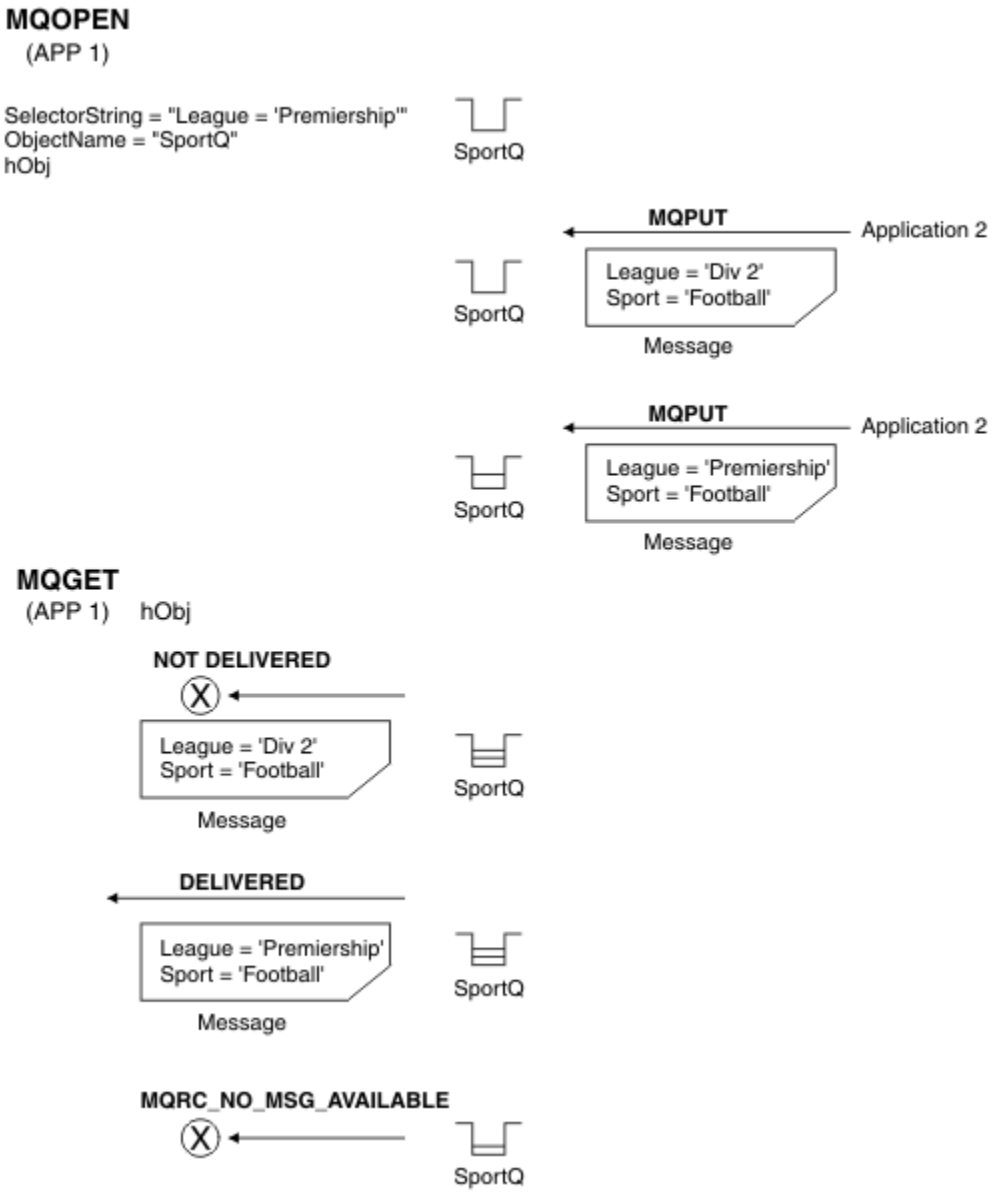
Obrázek 3 na stránce 31 zobrazuje proces výběru pomocí volání MQSUB.



Obrázek 3. Výběr pomocí volání MQSUB

Selektor lze předat při volání MQSUB pomocí pole *SelectionString* ve struktuře MQSD. Předání selektoru v rámci MQSUB má za následek, že v cílové frontě budou zpřístupněny pouze zprávy publikované do odebíraného tématu, které odpovídají zadanému řetězci výběru.

Obrázek 4 na stránce 32 zobrazuje proces výběru pomocí volání MQOPEN.



Obrázek 4. Výběr pomocí volání MQOPEN

Selektor lze předat ve volání MQOPEN pomocí pole *SelectionString* ve struktuře MQOD. Výsledkem předání selektoru ve volání MQOPEN je, že spotřebiteli zpráv jsou doručeny pouze zprávy v otevřené frontě, které odpovídají selektoru.

Hlavní použití selektoru ve volání MQOPEN je pro případ dvoubodového spojení, kdy aplikace může zvolit příjem pouze těch zpráv ve frontě, které odpovídají selektoru. Předchozí příklad ukazuje jednoduchý scénář, kdy jsou do fronty otevřené produktem MQOPEN vloženy dvě zprávy, ale aplikaci, která je získává, je přijata pouze jedna, protože se jedná o jedinou zprávu, která odpovídá selektoru.

Všimněte si, že následná volání MQGET vedou k MQRC_NO_MSG_AVAILABLE, protože ve frontě neexistují žádné další zprávy, které by odpovídaly danému selektoru.

Související pojmy

“Pravidla a omezení řetězce výběru” na stránce 39
Seznamte se s těmito pravidly o způsobu interpretace řetězců výběru a omezeních znaků, abyste se vyhnuli potenciálním problémům při používání selektorů.

Chování výběru

Přehled chování výběru IBM MQ .

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

- Má formát MQFMT_MD_EXTENSION
- Je bezprostředně následován platnou strukturou MQMDE.
- Jedná se o verzi 1 nebo obsahuje pouze výchozí pole verze 2.

Je možné, aby se řetězec výběru vyhodnocoval na hodnotu TRUE nebo FALSE před tím, než se provede jakákoli shoda s vlastnostmi zprávy. Může se jednat například o případ, kdy je řetězec výběru nastaven na hodnotu `"TRUE <> FALSE"`. Takové včasné vyhodnocení je zaručeno pouze v případě, že ve výběrovém řetězci nejsou žádné odkazy na vlastnosti zprávy.

Je-li řetězec výběru vyhodnocen na hodnotu TRUE před tím, než jsou vzaty v úvahu vlastnosti zprávy, jsou doručeny všechny zprávy publikované do tématu odebíraného spotřebitelem. Je-li řetězec výběru vyhodnocen jako FALSE před tím, než jsou zváženy vlastnosti zprávy, vrátí se kód příčiny MQRC_SELECTOR_ALWAYS_FALSE a kód dokončení MQCC_FAILED pro volání funkce, které prezentovalo selektor.

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

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

Výběr lze provést pouze na vlastnostech, které jsou přidruženy ke zprávě, nikoli na samotné zprávě, pokud není k dispozici poskytovatel rozšířeného výběru zpráv. Výběr lze na informačním obsahu zprávy provést pouze v případě, že je k dispozici poskytovatel rozšířeného výběru zpráv.

Ke každé vlastnosti zprávy je přidružen typ. Když provádíte výběr, musíte se ujistit, že hodnoty použité ve výrazech pro testování vlastností zprávy jsou správného typu. Dojde-li k neshodě typů, daný výraz se interpretuje jako FALSE.

Je vaší odpovědností zajistit, aby výběrové řetězce a vlastnosti zpráv používaly kompatibilní typy.

Kritéria výběru jsou i nadále používána pro neaktivní trvalé odběratele, takže jsou uchovány pouze zprávy, které odpovídají původně dodanému řetězci výběru.

Řetězce výběru nelze změnit při obnovení trvalého odběru pomocí příkazu alter (MQSO_ALTER). Pokud je při obnovení aktivity trvalého odběratele uveden jiný řetězec výběru, vrátí se aplikaci hodnota MQRC_SELECTOR_NOT_ALTERABLE.

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

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

Výkon systému zpráv

Výběr zpráv z fronty vyžaduje, aby produkt IBM MQ postupně kontroloval každou zprávu ve frontě. Zprávy jsou kontrolovány, dokud není nalezena zpráva, která odpovídá kritériím výběru, nebo nejsou k dispozici žádné další zprávy, které by bylo možné zkontrolovat. Proto je výkon systému zpráv náročný, pokud je výběr zpráv použit v hlubokých frontách.

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

- JMSCorrelationID = 'ID:*correlation_id*'

- JMSMessageID= 'ID:*ID_zprávy*'

kde:

- *correlation_id* je řetězec obsahující standardní IBM MQ identifikátor korelace.
- *message_id* je řetězec obsahující standardní IBM MQ identifikátor zprávy.

Poznámka: Selektor by měl odkazovat pouze na jednu z vlastností. Použití selektoru, který má jeden z těchto formátů, nabízí významné zlepšení výkonu při výběru JMSCorrelationID a nabízí mezní zlepšení výkonu pro JMSMessageID. Další informace viz téma “Selektory zpráv v adresáři JMS” na stránce 140.

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 závažné dopady na výkon a nadměrné požadavky na prostředky. Jako takový produkt IBM MQ ochrání systém tím, že nezpracuje příliš složité selektory, které by mohly vést k nedostatku systémových prostředků. Ochrana se může vyskytnout na výběrových řetězcích, které obsahují více než 100 testů, nebo když produkt IBM MQ zjistí, že se blíží omezení velikosti zásobníku operačního systému. Měli byste důkladně vyzkoušet a otestovat použití výběrových řetězců s mnoha komponentami na příslušných platformách, abyste se ujistili, že nejsou dosaženy limity ochrany.

Výkon a složitost selektorů lze zlepšit jejich zjednodušením pomocí dalších závorek pro kombinování komponent. Příklad:

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

Související pojmy

“Pravidla a omezení řetězce výběru” na stránce 39
Seznamte se s těmito pravidly o způsobu interpretace řetězců výběru a omezeních znaků, abyste se vyhnuli potenciálním problémům při používání selektorů.

Syntaxe selektoru zpráv

Selektor zpráv IBM MQ je řetězec se syntaxí, který je založen na podmnožině syntaxe podmíněného výrazu SQL92 .

Pořadí, ve kterém je selektor zpráv vyhodnocován, je zleva doprava v rámci úrovně priority. Chcete-li změnit toto pořadí, můžete použít závorky. Předdefinované literály selektoru a názvy operátorů jsou zde psány velkými písmeny, ale nerozlišují velká a malá písmena.

Pokud je selektor poskytován prostřednictvím rozhraní API, produkt IBM MQ ověří syntaktickou správnost selektoru zpráv v době, kdy je prezentován. Pokud je syntaxe řetězce výběru nesprávná nebo název vlastnosti není platný a poskytovatel rozšířeného výběru zpráv není k dispozici, aplikace se vrátí `MQRC_SELECTION_NOT_AVAILABLE` . Pokud je syntaxe řetězce výběru nesprávná nebo název vlastnosti není platný při obnovení odběru, vrátí se aplikaci hodnota `MQRC_SELECTOR_SYNTAX_ERROR` . Pokud bylo při nastavení vlastnosti zakázáno ověření názvu vlastnosti (nastavením `MQCMHO_NONE` namísto `MQCMHO_VALIDATE`) a aplikace následně vloží zprávu s neplatným názvem vlastnosti, tato zpráva se nikdy nevybere.

V době, kdy je selektor prezentován, není vrácena žádná chyba, pokud produkt IBM MQ zjistí, že administrativně definovaný selektor odběrů používá rozšířenou syntaxi zpráv, jak uvádí parametr **`DISPLAY SUB SELTYPE`** s hodnotou `EXTENDED`. V tomto případě je kontrola syntaxe řetězce výběru odložena do doby publikování (viz `MQRC_SELECTION_NOT_AVAILABLE`).

Selektor může obsahovat:

- Literály:
  - Řetězcové literály jsou uzavřeny v apostrofech. Dvě po sobě jdoucí jednoduché uvozovky představují jednoduchou uvozovku. Příklady jsou 'literal' a 'literal'. Stejně jako řetězcové literály Java používají

kódování znaků Unicode. K uzavření řetězcového literálu nelze použít dvojité uvozovky. Mezi jednoduchými uvozovkami lze použít libovolnou posloupnost bajtů.

– Bajtový řetězec je jedna nebo více dvojic hexadecimálních znaků uzavřených v uvozovkách a s předponou `0x`. Příklady jsou `"0x2F1C"` nebo `"0XD43A"`. Délka bajtového řetězce musí být alespoň jeden bajt. Pokud se bajtový řetězec selektoru shoduje s vlastností zprávy typu `MQTYPE_BYTE_STRING`, neprovede se žádná speciální akce na úvodní nebo koncové nule. S bajty se zachází jako s jiným znakem. Endianness se také nepovažuje za. Délka řetězce bajtů selektoru i vlastnosti musí být stejná a posloupnost bajtů musí být stejná.

  Příklady výběrů bajtových řetězců (předpokládejme *myBytes* = `0AFC23`), které odpovídají:

  - `"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 v porovnání významná)
  - `"myBytes = "0x000AFC23"" = FALSE` (protože úvodní nula je v porovnání významná)
  - `"myBytes = "0x23FC0A"" = FALSE` (protože endianness není považován za)

– Hexadecimální čísla začínají nulou a následují velká nebo malá písmena x. Zbytek literálu obsahuje jeden nebo více platných hexadecimálních znaků. Příklady jsou `0xA`, `0xAF`, `0X2020`.

– Úvodní nula následovaná jednou nebo více číslicemi v rozsahu 0-7 je vždy interpretována jako začátek osmičkové číslice. Nemůžete reprezentovat desetinné číslo s nulovou předponou, jako je toto, například `09` vrací chybu syntaxe, protože 9 není platná osmičková číslice. Příklady osmičkových čísel jsou `0177`, `0713`.

– Přesný číselný literál je číselná hodnota bez desetinné čárky, například 57, `-957a` `+62`. Přesný číselný literál může mít koncové velké nebo malé písmeno 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 číselný literál je číselná hodnota ve vědecké notaci, například `7E3` nebo `-57.9E2`, nebo číselná hodnota s desetinným číslem, například `7.`, `-95.7`nebo `+6.2`. Produkt IBM MQ podporuje čísla v rozsahu `-1.797693134862315E+308` až `1.797693134862315E+308`.

  Význam by měl následovat za volitelným znakem (+ nebo -). Význam by měl být buď celé číslo, nebo zlomek. Zlomková část významnosti a nemusí mít úvodní číslici.

  Velká nebo malá písmena E označují začátek volitelného exponentu. Exponent má desetinný radix a číselná část exponentu může mít předponu volitelným znakovým znakem.

  Přibližné číselné literály mohou být ukončeny znakem F nebo D (bez rozlišování malých a velkých písmen). Tato syntaxe existuje pro podporu křížové metody značení čísel s jednoduchou nebo dvojitou přesností. Tyto znaky jsou volitelné a neovlivňují způsob uložení nebo zpracování přibližného číselného literálu. Tato čísla jsou vždy uložena a zpracována s dvojitou přesností.

– Logické literály `TRUE` a `FALSE`.

Poznámka: Nekonečné reprezentace IEEE-754 , jako např. `NaN`, `+Infinity`, `-Infinity` , nejsou ve výběrových řetězcích podporovány. Proto není možné použít tyto hodnoty jako operandy ve výrazu. Záporná nula je pro matematické operace považována za kladnou nulu.

- Identifikátory:

  Identifikátor je posloupnost znaků s proměnnou délkou, která musí začínat platným počátečním znakem identifikátoru, za nímž následuje nula nebo více platných znaků části identifikátoru. Pravidla pro názvy identifikátorů jsou stejná jako pro názvy vlastností zpráv, další informace viz “Názvy vlastností” na stránce 27 a “Omezení názvu vlastnosti” na stránce 28 .

  Poznámka: Výběr lze na informačním obsahu zprávy provést pouze v případě, že je k dispozici poskytovatel rozšířeného 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, i když se provádí číselné povýšení tam, kde je to možné. Pokud dojde k neshodě typů, výsledek výrazu je FALSE. Pokud je odkazována vlastnost, která ve zprávě neexistuje, její hodnota je NULL.

Převody typů, které se vztahují na metody get pro vlastnosti, se nepoužijí, pokud je vlastnost použita ve výrazu selektoru zpráv. Nastavíte-li například vlastnost jako řetězcovou hodnotu a poté použijete selektor k dotazování jako číselnou hodnotu, výraz vrátí hodnotu FALSE.

Pole JMS a názvy vlastností, které se mapují na názvy vlastností nebo názvy polí MQMD , jsou také platnými identifikátory ve výběrovém řetězci. IBM MQ mapuje rozpoznané pole JMS a názvy vlastností na hodnoty vlastností zprávy. Další informace viz “Selektory zpráv v adresáři JMS” na stránce 140. Jako příklad výběrový řetězec "JMSPriority >=" vybere vlastnost Pri nalezenou ve složce jms aktuální zprávy.

- Přetečení/podtečení:

  Pro desetinná i přibližná číselná čísla nejsou definovány následující podmínky:

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

  Pro tyto podmínky se neprovádějí žádné kontroly.

- Netisknutelné:

  Definuje se jako mezera, posuv na nový řádek, znak CR, vodorovný tabulátor nebo svislý tabulátor. Následující znaky Unicode jsou rozpoznány jako mezery:

  – \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 vyhodnotí jako pravdivý, se shoduje; selektor, který se vyhodnotí jako nepravdivý nebo neznámý, se neshoduje.
  – Aritmetické výrazy se skládají ze sebe, aritmetických operací, identifikátorů (hodnota identifikátoru je považována za číselný literál) a číselných literálů.
  – Podmíněné výrazy se skládají ze sebe samých, operací porovnání a logických operací.

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

  – Dva bajtové řetězce se rovnají pouze v případě, že jsou řetězce stejné délky a pořadí bajtů je stejné.

  - Porovnávat lze pouze hodnoty stejného typu. Jednou z výjimek je, že je platné porovnat přesné číselné hodnoty a přibližné číselné hodnoty (požadovaný převod typu je definován pravidly Java číselné povýšení). Pokud dojde k pokusu o porovnání různých typů, selektor bude mít vždy hodnotu false.
  - Porovnání řetězců a logických hodnot je omezeno na = a <>. Dva řetězce jsou stejné pouze v případě, že obsahují stejnou posloupnost znaků.
- Aritmetické operátory v pořadí podle priority:
  - +, - unární.
  - * násobení a / dělení.
  - + sčítání a - odečítání.
  - Aritmetické operace s hodnotou NULL nejsou podporovány. Pokud se o ně pokusíte, úplný selektor bude mít vždy hodnotu false.
  - Aritmetické operace musí používat číselné povýšení Java .
- arithmetic-expr1 [NOT] BETWEEN arithmetic-expr2 a arithmetic-expr3 porovnávací operátor:
  - Hodnota `Age BETWEEN 15 and 19` je ekvivalentní hodnotě `age>= 15 AND age <= 19`.
  - Hodnota `Age NOT BETWEEN 15 and 19` je ekvivalentní hodnotě `age < 15 OR age > 19`.
  - Pokud má některý z výrazů operace BETWEEN hodnotu NULL, hodnota operace je false. Pokud je některý z výrazů operace `NOT BETWEEN` NULL, hodnota operace je true.
- identifikátor `[NOT] IN (string-literal1, string-literal2,...)` porovnávací operátor, kde identifikátor má hodnotu String nebo NULL .
  - `Country IN ('UK', 'US', 'France')` je true pro `'UK'` a false pro `'Peru'`. Je ekvivalentem výrazu `(Country = 'UK') OR (Country = 'US') OR (Country = 'France')`.
  - `Country NOT IN ('UK', 'US', 'France')` je false pro `'UK'` a true pro `'Peru'`. Je ekvivalentem výrazu `NOT ((Country = 'UK') OR (Country = 'US') OR (Country = 'France'))`.
  - Pokud je identifikátor operace IN nebo `NOT IN` NULL, hodnota operace je neznámá.
- `identifier [NOT] LIKE` *`pattern-value`* `[ESCAPE` *`escape-character`* `]` operátor porovnání, kde `identifier` má hodnotu řetězce. *pattern-value* je řetězcový literál, kde _ představuje libovolný jednotlivý znak a % představuje libovolnou posloupnost znaků (včetně prázdné posloupnosti). Všechny ostatní znaky stojí za sebou. Volitelný *řídicí znak* je jednoznakový řetězcový literál, který se používá jako řídicí znak pro speciální význam znaků _ a % in *pattern-value*. Operátor LIKE musí být použit pouze k porovnání dvou řetězcových hodnot.
  - `phone LIKE '12%3'` platí pro 123 a 12993 a false pro 1234.
  - `word LIKE 'l_se'` platí 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.
  - Pokud je identifikátor operace LIKE nebo `NOT LIKE` NULL, hodnota operace je neznámá.

  Poznámka: K porovnání dvou řetězcových hodnot musí být použit operátor LIKE . Hodnota `Root.MQMD.CorrelId` je 24bajtové bajtové pole, nikoli znakový řetězec. Řetězec selektoru `Root.MQMD.CorrelId LIKE 'ABC%'` je analyzátorem přijat jako syntakticky platný, ale je vyhodnocen jako false. Při porovnávání bajtového pole se znakovým řetězcem proto nelze použít řetězec LIKE .
- `identifier IS NULL` porovnávací operátor testuje hodnotu pole záhlaví NULL nebo chybějící hodnotu vlastnosti.
- `identifier IS NOT NULL` porovnávací operátor testuje existenci nenulové hodnoty pole záhlaví 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 vede k neznámé hodnotě.
– Operátory IS NULL a IS NOT NULL převádějí neznámou hodnotu na hodnoty TRUE a FALSE .

Logické operátory používají logiku se třemi hodnotami (T=TRUE, F=FALSE, U=UNKNOWN).

Tabulka 1. Hodnota výsledku logického operátoru, když je logika A AND B

Operátor A	**Provozovatel B**	**Výsledek (A A A A 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. Hodnota výsledku logického operátoru, když je logika A OR B

Operátor A	**Provozovatel 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. Hodnota výsledku logického operátoru, když je logika NOT A

Operátor A	**Výsledek (NOT A)**
T	F
F	T
U	U

Následující selektor zpráv vybírá zprávy s typem zprávy auto, modrou barvou a hmotností větší než 2500 lbs:

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

Ačkoli SQL podporuje pevné desetinné porovnání a aritmetiku, selektory zpráv ne. To je důvod, proč jsou přesné číselné literály omezeny na ty, které nemají desetinné číslo. To je také důvod, proč existují číslice s desetinným číslem jako alternativní reprezentace pro přibližnou číselnou hodnotu.

Komentáře SQL nejsou podporovány.

Související pojmy

"Vlastnosti zprávy" na stránce 26
Pomocí vlastností zprávy můžete aplikaci povolit výběr zpráv ke zpracování nebo načtení informací o zprávě bez přístupu k záhlavím MQMD nebo MQRFH2 . Také usnadňují komunikaci mezi aplikacemi IBM MQ a JMS .

Související odkazy

MsgHandle

MQBUFMH-Převést vyrovnávací paměť na manipulátor zprávy

Pravidla a omezení řetězce výběru

Seznamte se s těmito pravidly o způsobu interpretace řetězců výběru a omezeních znaků, abyste se vyhnuli potenciálním problémům při používání selektorů.

- Výběr zprávy pro systém zpráv publikování/odběru se vyskytuje ve zprávě odeslané vydavatelem. Viz řetězce výběru.
- Ekvivalence se testuje pomocí jediného znaku rovná se; například `a = b` je správné, zatímco `a == b` je nesprávné.
- Operátor používaný mnoha programovacími jazyky ke znázornění 'not equal to' je `!=`. Tato reprezentace není platným synonymem pro <> ; například `a <> b` je platný, zatímco `a != b` není platný.
- Jednoduché uvozovky jsou rozpoznány pouze v případě, že ' Použije se znak (U+0027). Podobně dvojité uvozovky, platné pouze v případě, že jsou použity k uzavření bajtových řetězců, musí používat " (U+0022) znak.
- Symboly &, &&, | a || nejsou synonyma pro logické spojení/odpojení; například `a && b` musí být zadáno jako `a AND b`.
- Zástupné znaky * a ? nejsou synonyma pro % a _.
- Selektory obsahující složené výrazy, jako např. `20 < b < 30` , nejsou platné. Syntaktický analyzátor vyhodnocuje operátory, které mají stejnou prioritu zleva doprava. Příklad by se tedy stal `(20 < b) < 30`, což nedává smysl. Místo toho musí být výraz zapsán jako `(b > 20) AND (b < 30)`.
- Bajtové řetězce musí být uzavřeny do dvojitých uvozovek; pokud se používají jednoduché uvozovky, bajtový řetězec se pokládá za řetězcový literál. Počet znaků (nikoli počet, který tyto znaky představují) za položkou `0x` musí být násobkem dvou.
- Klíčové slovo `IS` není synonymem pro znak rovná se. Proto nejsou řetězce výběru `a IS 3` a `b IS 'red'` platné. Klíčové slovo `IS` existuje pouze pro podporu případů `IS NULL` a `IS NOT NULL` .

Související pojmy

"Chování výběru" na stránce 33
Přehled chování výběru IBM MQ .

Související odkazy

Řetězce výběru

Aspekty kódování UTF-8 a kódování Unicode při použití selektorů zpráv

Znaky, které nejsou uzavřeny v apostrofech a které tvoří vyhrazená klíčová slova řetězce výběru, musí být zadány v kódování Basic Latin Unicode (v rozsahu od znaku U+0000 do U+0007F). Není platné použít jiné reprezentace kódových bodů alfanumerických znaků. Například číslo 1 musí být vyjádřeno jako U+0031 v Unicode, není platné použít ekvivalent číslice celé šířky U+FF11 nebo ekvivalent arabštiny U+0661.

Názvy vlastností zpráv lze zadat pomocí libovolné platné posloupnosti znaků Unicode. Názvy vlastností zpráv obsažené v řetězcích výběru, které jsou zakódovány v UTF-8 , budou ověřeny i v případě, že obsahují vícebajtové znaky. Ověření vícebajtové znakové sady UTF-8 je striktní a musíte se ujistit, že pro názvy vlastností zpráv jsou použity platné posloupnosti UTF-8 . Znaky mimo základní vícejazyčnou

rovinu Unicode (výše uvedené znaky U + FFFF) reprezentované v kódu UTF-16 náhradními kódovými body (X'D800'až X'DFFF') nebo čtyřmi bajty v kódování UTF-8nejsou v názvech vlastností zpráv podporovány.

Při porovnávání za účelem dosažení shody se na názvech vlastností nebo hodnotách neprovádí žádné další zpracování. To například znamená, že se neprovádí žádná předkompozice a ligatury nemají žádný zvláštní význam. Například předem sestavený přečíslovaný znak U+00FC není považován za ekvivalentní znaku U+0075 + U+0308 a hodnota ff posloupnosti znaků není považována za ekvivalentní znaku Unicode U+FB00 (LATIN SMALL LIGATURE)

Data vlastností uzavřená v apostrofech mohou být reprezentována libovolnou posloupností bajtů a nejsou ověřena.

Výběr obsahu zprávy

Je možné se přihlásit k odběru na základě výběru obsahu informačního obsahu zprávy (také známého jako filtrování obsahu), ale rozhodnutí, které zprávy mají být doručeny do takového odběru, nemůže být provedeno přímo produktem IBM MQ; místo toho je pro zpracování zpráv vyžadován poskytovatel rozšířeného výběru zpráv, například IBM Integration Bus.

Když aplikace publikuje v řetězci tématu, kde jeden nebo více odběratelů má výběrový řetězec vybraný v obsahu zprávy, produkt IBM MQ vyžádá, aby poskytovatel rozšířeného výběru zpráv analyzoval publikování a informoval produkt IBM MQ , zda publikování odpovídá kritériím výběru určeným každým odběratelem s filtrem obsahu.

Pokud poskytovatel rozšířeného výběru zpráv určí, že publikování odpovídá řetězci výběru odběratele, bude zpráva i nadále doručována odběrateli.

Pokud poskytovatel rozšířeného výběru zpráv zjistí, že se publikování neshoduje, zpráva nebude doručena odběrateli. To může způsobit selhání volání MQPUT nebo MQPUT1 s kódem příčiny MQRC_PUBLICATION_FAILURE. Pokud poskytovatel rozšířeného výběru zpráv nemůže publikaci analyzovat, vrátí se kód příčiny MQRC_CONTENT_ERROR a volání MQPUT nebo MQPUT1 se nezdaří.

Pokud poskytovatel rozšířeného výběru zpráv není k dispozici nebo nemůže určit, zda má odběratel obdržet publikování, vrátí se kód příčiny MQRC_SELECTION_NOT_AVAILABLE a volání MQPUT nebo MQPUT1 se nezdaří.

Při vytváření odběru s filtrem obsahu a nedostupnosti poskytovatele rozšířeného výběru zpráv dojde k selhání volání MQSUB s kódem příčiny MQRC_SELECTION_NOT_AVAILABLE. Pokud se obnovuje odběr s filtrem obsahu a poskytovatel rozšířeného výběru zpráv není k dispozici, volání MQSUB vrátí varování MQRC_SELECTION_NOT_AVAILABLE, ale odběr je možné obnovit.

Související odkazy

Řetězce výběru

Asynchronní spotřeba zpráv IBM MQ

Asynchronní spotřeba používá sadu rozšíření rozhraní MQI (Message Queue Interface), MQI volá MQCB a MQCTL, které umožňují zápis aplikace MQI pro příjem zpráv ze sady front. Zprávy jsou doručeny aplikaci vyvoláním 'jednotky kódu', identifikované aplikací, která předává zprávu, nebo tokenem, který představuje zprávu.

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

V asynchronní spotřebě zpráv se používají následující výrazy:

Spotřebitel zpráv.
Programovací konstrukt, který umožňuje definovat program nebo funkci, které mají být vyvolány se zprávou, když je k dispozici ten, který odpovídá požadavkům aplikace.

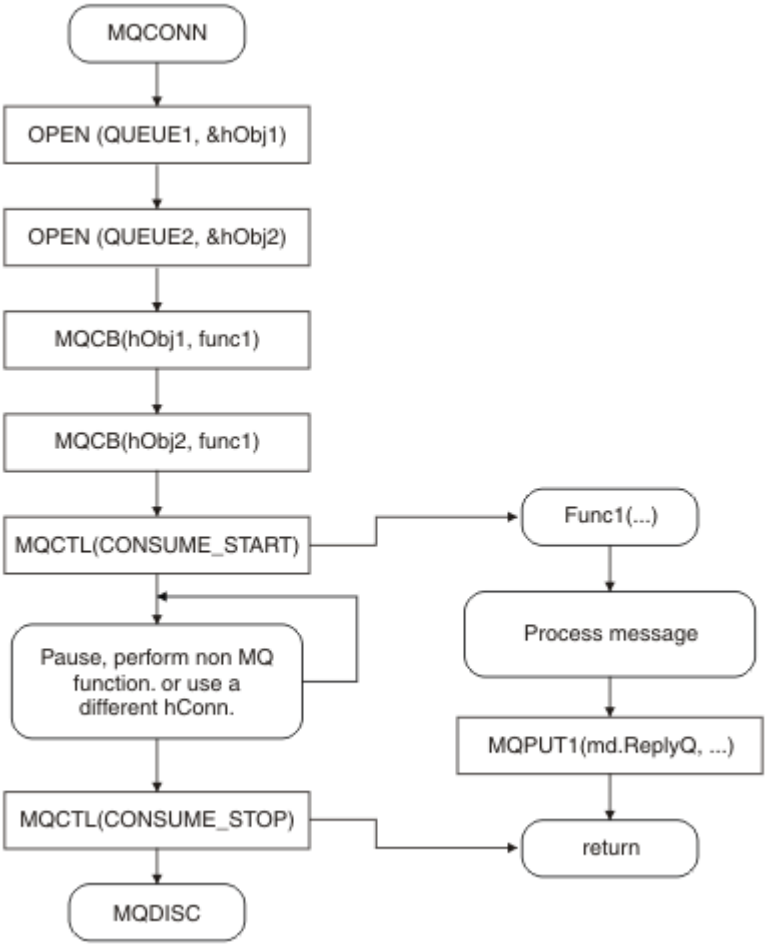
obslužná rutina událostí
Programovací konstrukt, který umožňuje definovat program nebo funkci, která má být vyvolána při výskytu asynchronní události, například při uvedení správce front do klidového stavu.

Zpětné volání
Generický výraz používaný k odkazování na rutinu spotřebitele zpráv 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 zpracováváte zprávy v pořadí priorit, je pořadí priorit v každé frontě sledováno nezávisle: zprávy s nízkou prioritou můžete získat z jedné fronty před zprávami s vysokou prioritou z druhé. Pořadí zpráv ve více frontách není zaručeno. Všimněte si také, že pokud používáte uživatelské procedury rozhraní API, možná je budete muset změnit tak, aby zahrnovaly volání MQCB a MQCTL.

Následující ilustrace poskytují příklad použití této funkce.

Obrázek 5 na stránce 41 zobrazuje aplikaci s podporou podprocesů, která spotřebovává zprávy ze dvou front. Příklad zobrazuje všechny zprávy doručované do jediné funkce.

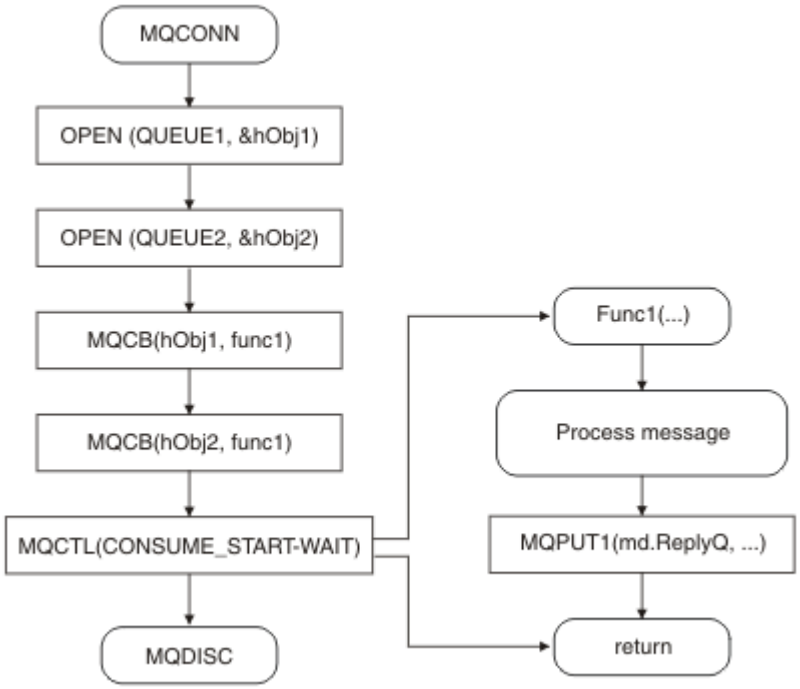


Obrázek 5. Standardní aplikace řízená zprávami spotřebovávající ze dvou front

z/OS V systému z/OSmusí hlavní řídicí podproces před ukončením zadat volání MQDISC. To umožňuje, aby všechny podprocesy zpětného volání ukončily a uvolňovaly systémové prostředky.

Obrázek 6 na stránce 42 Tento ukázkový tok zobrazuje jednu aplikaci s podporou podprocesů, která spotřebovává zprávy ze dvou front. Příklad zobrazuje všechny zprávy doručované do jediné funkce.

Rozdíl od asynchronního případu spočívá v tom, že řízení se nevrátí vydavateli MQCTL, dokud se všichni spotřebitelé sami nedeaktivují; to znamená, že jeden spotřebitel vydal požadavek MQCTL STOP nebo správce front neuvede do klidového stavu.



Obrázek 6. Aplikace řízená zprávami s jedním podprocesem, která spotřebovává ze dvou front

Skupiny zpráv

Zprávy se mohou vyskytovat ve skupinách, aby se umožnilo řazení zpráv.

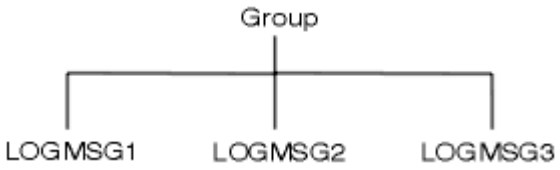
Skupiny zpráv umožňují, aby bylo více zpráv označeno jako vzájemně související, a logické pořadí, které se má použít na skupinu (viz "Logické a fyzické řazení" na stránce 749). V systému Multiplatforms segmentace zpráv umožňuje rozdělení velkých zpráv do menších segmentů. Při vkládání do tématu nelze používat seskupené ani segmentované zprávy.

Hierarchie ve skupině je následující:

Skupina
Jedná se o nejvyšší úroveň v hierarchii a je identifikována pomocí *GroupId*. Skládá se z jedné nebo více zpráv, které obsahují stejnou zprávu *GroupId*. Tyto zprávy lze uložit kdekoli ve frontě.

Poznámka: Výraz *zpráva* se zde používá k označení jedné položky ve frontě, například by byl vrácen jedním příkazem MQGET, který neurčuje MQGMO_COMPLETE_MSG.

Obrázek 7 na stránce 42 zobrazuje skupinu logických zpráv:



Obrázek 7. Skupina logických zpráv

Otevřením fronty a zadáním MQOO_BIND_ON_GROUP vynutíte, aby byly všechny zprávy ve skupině odeslané do této fronty odeslány do stejné instance fronty. Další informace o volbě BIND_ON_GROUP naleznete v tématu Zpracování afinit zpráv.

Logická zpráva
Logické zprávy ve skupině jsou identifikovány pomocí polí *GroupId* a *MsgSeqNumber* . *MsgSeqNumber* začíná na 1 pro první zprávu ve skupině, a pokud zpráva není ve skupině, hodnota pole je 1.

Pomocí logických zpráv v rámci skupiny můžete:

- Zajistit objednávání (pokud to není zaručeno za okolností, za kterých je zpráva přenášena).
- Povolit aplikacím seskupovat podobné zprávy (například ty, které musí všechny zpracovat stejná instance serveru).

Každá zpráva ve skupině se skládá z jedné fyzické zprávy, pokud není rozdělena na segmenty. Každá zpráva je logicky samostatnou zprávou a pouze pole *GroupId* a *MsgSeqNumber* v deskriptoru MQMD musí mít jakýkoli vztah k ostatním zprávám ve skupině. Ostatní pole v deskriptoru MQMD jsou nezávislá; některá mohou být identická pro všechny zprávy ve skupině, zatímco jiná mohou být odlišná. Například zprávy ve skupině mohou mít různé názvy formátů, CCSID a kódování.

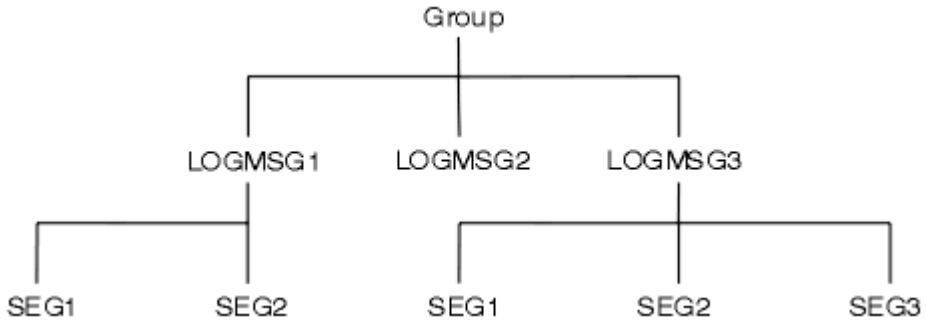
Segment

Segmenty se používají ke zpracování zpráv, které jsou příliš velké pro vkládající nebo získávanou aplikaci nebo správce front (včetně intervenujících správců front, jejichž prostřednictvím zpráva prochází). Další informace viz “Segmentace zpráv” na stránce 767.

Jednotlivá zpráva je rozdělena do menších zpráv nazývaných *segmenty*. Segment zprávy je identifikován poli *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 43 zobrazuje příklad zpráv ve skupině). Segment je logicky součástí jediné zprávy, takže pouze pole *MsgId*, *Offset*a *MsgFlags* v deskriptoru MQMD by se měla v jednotlivých segmentech téže zprávy lišit. Pokud segment nedorazí, vrátí se kód příčiny MQRC_INCOMPLETE_GROUP nebo MQRC_INCOMPLETE_MSG jako odpovídající.

Obrázek 8 na stránce 43 zobrazuje skupinu logických zpráv, z nichž některé jsou segmentované:



Obrázek 8. Segmentované zprávy

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

Nemůžete použít segmentované nebo seskupené zprávy s publikováním/odběrem.

Související pojmy

“Segmentace zpráv” na stránce 767
Pomocí těchto informací získáte informace o segmentaci zpráv. Tato funkce není podporována v systému IBM MQ for z/OS nebo aplikacemi používajícími IBM MQ classes for JMS.

Související odkazy

“Logické a fyzické řazení” na stránce 749
Zprávy ve frontách se mohou vyskytovat (v rámci každé úrovně priority) v *fyzickém* nebo *logickém* pořadí.

MQMD-Deskriptor zpráv

Trvalost zpráv

Trvalé zprávy se zapisují do protokolů a datových souborů fronty. Pokud je správce front restartován po selhání, obnoví tyto trvalé zprávy podle potřeby z protokolovaných dat. Zprávy, které nejsou trvalé, jsou vyřazeny v případě, že se správce front zastaví, bez ohledu na to, zda je zastavení výsledkem příkazu operátora, nebo kvůli selhání některé části systému.

z/OS Výjimkou jsou přechodné zprávy uložené v prostředku Coupling Facility (CF) v systému z/OS . Trvají tak dlouho, dokud je prostředek CF k dispozici.

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. Volitelně můžete nastavit perzistenci zprávy pomocí pole *Persistence* struktury MQMD, abyste definovali zprávu jako trvalou nebo dočasnou.

Výkon aplikace je ovlivněn při použití trvalých zpráv; rozsah účinku závisí na charakteristice výkonu subsystému I/O počítače a způsobu použití voleb synchronizačního bodu na každé platformě:

- Trvalá zpráva, mimo aktuální jednotku práce, je zapsána na disk při každé operaci vložení a získání. Viz “Potvrzení a zálohování jednotek práce” na stránce 825.
- z/OS ALW Pro všechny platformy kromě platformy IBM ije trvalá zpráva v rámci aktuální transakce protokolována pouze v případě, že je transakce potvrzena a jednotka práce může obsahovat mnoho operací fronty.

Přechodné zprávy lze použít pro rychlé zasílání zpráv. Další informace o rychlých zprávách naleznete v tématu Bezpečnost zpráv .

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

Zprávy, které se nepodařilo doručit

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

Můžete provést následující akce:

- Pokuste se znovu vložit zprávu do fronty.
- Požadavek na vrácení zprávy odesilateli.
- Vložte zprávu do fronty nedoručených zpráv.

Další informace viz“Zpracování chyb procedurálních programů” na stránce 1004.

Zprávy, které jsou vráceny zpět

Při zpracování zpráv z fronty pod kontrolou jednotky práce se může jednotka práce skládat z jedné nebo více zpráv. Dojde-li k vrácení, zprávy, které byly načteny z fronty, jsou znovu uvedeny do původního stavu ve frontě a lze je znovu zpracovat v jiné pracovní jednotce. Pokud zpracování konkrétní zprávy způsobuje problém, jednotka práce je znovu vrácena zpět. To může způsobit smyčku zpracování. Zprávy, které byly vloženy do fronty, jsou z fronty odebrány.

Aplikace může zjistit zprávy, které jsou v takovém cyklu zachyceny, testováním pole *BackoutCount* deskriptoru MQMD. Aplikace může buď opravit situaci, nebo vydat varování operátorovi.

Multi Počet vrácení vždy přežije restartování správce front. Jakákoli změna atributu **HardenGetBackout** se ignoruje.

z/OS V případě sdílených front počet vrácení vždy přežije restartování správce front. Chcete-li pro všechny ostatní konfigurace v systému z/OSzajistit, aby počet vrácení pro soukromé fronty přežil restartování správce front, nastavte atribut *HardenGetBackout* na hodnotu MQQA_BACKOUT_HARDENED; jinak, pokud má být správce front restartován, nebude udržovat přesný počet vrácení pro každou zprávu. Nastavení atributu tímto způsobem přidá náklady na další zpracování.

Další informace o potvrzování a zálohování zpráv viz “Potvrzení a zálohování jednotek práce” na stránce 825.

Fronta a správce front pro odpovědi

Existují případy, kdy můžete obdržet zprávy v reakci na zprávu, kterou odešlete:

- Zpráva odpovědi jako odpověď na zprávu požadavku.
- Zpráva sestavy o neočekávané události nebo vypršení platnosti
- Zpráva o události COA (Confirmation Of Arrival) nebo COD (Confirmation Of Delivery)
- Zpráva sestavy o události PAN (Positive Action Notification) nebo NAN (Negative Action Notification)

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

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

ReplyToQ
Pokud je *ReplyToQ* lokální definicí vzdálené fronty, pole *ReplyToQ* se nastaví na název vzdálené fronty; jinak se toto pole nezmění.

ReplyToQMgr
Pokud je *ReplyToQ* lokální definicí vzdálené fronty, pole *ReplyToQMgr* je nastaveno na název správce front, který vlastní vzdálenou frontu; jinak je pole *ReplyToQMgr* nastaveno na název správce front, ke kterému je vaše aplikace připojena.

Poznámka: Můžete požadovat, aby správce front provedl více než jeden pokus o doručení zprávy, a můžete požadovat, aby byla zpráva vyřazena, pokud se nezdaří. Nemá-li být zpráva po neúspěšném doručení zahozena, vzdálený správce front vloží zprávu do fronty nedoručených zpráv (viz "Použití fronty nedoručených zpráv (nedoručených zpráv)" na stránce 1007).

kontext zprávy

Kontext zprávy umožňuje aplikaci, která načte zprávu, zjistit informace o původci zprávy.

Načtení aplikace může chtít:

- Zkontrolujte, zda má odesílající aplikace správnou úroveň oprávnění.
- Provést nějakou funkci evidence, aby mohla účtovat odesílající aplikaci za jakoukoli práci, kterou musí provést.
- Uchovat záznam pro audit všech zpráv, se kterými pracoval

Při použití volání MQPUT nebo MQPUT1 k vložení zprávy do fronty můžete určit, že správce front má do deskriptoru zprávy přidat některé výchozí informace o kontextu. Aplikace, které mají odpovídající úroveň oprávnění, mohou přidat další informace o kontextu. Další informace o tom, jak zadat informace o kontextu, viz "Řízení informací o kontextu zprávy" na stránce 736.

Uživatelský kontext je používán správcem front při generování následujících typů zpráv sestavy:

- Potvrdit při doručení
- Vypršení

Když jsou tyto zprávy sestavy generovány, je kontext uživatele zkontrolován pro oprávnění + put a + passid na místě určení sestavy. Pokud má uživatelský kontext nedostatečné oprávnění, zpráva sestavy se umístí do fronty nedoručených zpráv, pokud byla definována. Pokud neexistuje žádná fronta nedoručených zpráv, zpráva sestavy se zahodí.

Všechny informace o kontextu jsou uloženy v polích kontextu deskriptoru zprávy. Typ informací spadá do informací o identitě, původu a kontextu uživatele.

kontext identity

Informace o *kontextu identity* identifikují uživatele aplikace, který nejprve vložil zprávu do fronty. Vhodně autorizované aplikace mohou nastavit následující pole:

- Správce front vyplní pole *UserIdentifier* názvem, který identifikuje uživatele. Způsob, jakým to může správce front provést, závisí na prostředí, ve kterém je aplikace spuštěna.

- Správce front vyplní pole *AccountingToken* tokenem nebo číslem určeným z aplikace, která vložila zprávu.
- Aplikace mohou použít pole *ApplIdentityData* pro jakékoli další informace, které chtějí zahrnout o uživateli (například šifrované heslo).

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

Informace o tom, jak správce front vyplňuje 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 také měly předávat informace o kontextu identity, aby ostatní aplikace znaly identitu původce zprávy.

Původní kontext

Původní kontext popisuje aplikaci, která vložila zprávu do fronty, ve které je zpráva aktuálně uložena. Deskriptor zprávy obsahuje následující pole pro informace o původním kontextu:

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

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

Aplikace s dostatečným oprávněním může poskytnout svůj vlastní kontext. To umožňuje zachovat informace o monitorování účtů, když má jeden uživatel jiné ID uživatele na každém ze systémů, které zpracovávají zprávu, ze které pocházejí.

IBM MQObjekty

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

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

Pokud například pomocí příkazů IBM MQ definujete, měníte nebo odstraňujete objekty, správce front zkontroluje, zda máte požadovanou úroveň oprávnění k provádění těchto operací. Podobně platí, že pokud aplikace používá k otevření objektu volání MQOPEN, správce front zkontroluje, zda má aplikace požadovanou úroveň oprávnění, než povolí přístup k danému objektu. Kontroly se provádějí na názvu otevíraného objektu.

Související pojmy

“Řízení informací o kontextu zprávy” na stránce 736
Při použití volání MQPUT nebo MQPUT1 k vložení zprávy do fronty můžete určit, že správce front má do deskriptoru zprávy přidat některé výchozí informace o kontextu. Aplikace, které mají odpovídající úroveň oprávnění, mohou přidat další informace o kontextu. Pole voleb ve struktuře MQPMO můžete použít k řízení informací o kontextu.

Související odkazy

“Volby MQOPEN týkající se kontextu zprávy” na stránce 727

Chcete-li mít možnost přidružit informace o kontextu ke zprávě při jejím vložení do fronty, musíte při otevření fronty použít jednu z voleb kontextu zprávy.

Windows Příprava a spuštění aplikací serveru Microsoft Transaction Server

Chcete-li připravit aplikaci MTS ke spuštění jako aplikace IBM MQ MQI client , postupujte podle těchto pokynů, které jsou vhodné pro vaše prostředí.

Obecné informace o vývoji aplikací Microsoft Transaction Server (MTS), které přistupují k prostředkům systému IBM MQ , naleznete v části MTS v Centru nápovědy IBM MQ .

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

- Pokud komponenta používá vazby jazyka C pro rozhraní MQI, postupujte podle pokynů v souboru "Příprava programů v jazyce C v adresáři Windows" na stránce 983 , ale propojte komponentu s knihovnou mqicxa.lib namísto mqic.lib.
- Pokud komponenta používá třídy IBM MQ C++, postupujte podle pokynů v části "Sestavení programů C++ na Windows" na stránce 535 , ale propojte komponentu s knihovnou imqx23vn.lib místo imqc23vn.lib.
- Pokud komponenta používá vazby jazyka Visual Basic pro rozhraní MQI, postupujte podle pokynů v části "Příprava programů Visual Basic v adresáři Windows" na stránce 986 , ale při definování projektu Visual Basic zadejte do pole **Argumenty podmíněné kompilace** hodnotu `MqType=3` .

Aspekty návrhu pro aplikace IBM MQ

Když jste se rozhodli, jak mohou vaše aplikace využívat platformy a prostředí, která máte k dispozici, musíte se rozhodnout, jak používat funkce nabízené produktem IBM MQ.

Při návrhu aplikace IBM MQ zvažte následující otázky a volby:

Druh žádosti

Jaký je účel vaší žádosti? Informace o různých typech aplikací, které můžete vyvíjet, naleznete v následujících odkazech:

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

Kromě toho můžete také napsat vlastní aplikace pro automatizaci administrace produktu IBM MQ. Další informace viz Rozhraní administrace IBM MQ (MQAI) a Automatizace administrativních úloh.

Programovací jazyk

Produkt IBM MQ podporuje pro psaní aplikací řadu různých programovacích jazyků. Další informace viz "Vyvíjení aplikací pro IBM MQ" na stránce 5.

Aplikace pro více než jednu platformu

Bude vaše aplikace spuštěna na více než jedné platformě? Máte strategii, jak se přesunout na jinou platformu, než tu, kterou dnes používáte? Je-li odpověď na některou z těchto otázek ano, ujistěte se, že kód své programy pro nezávislost platformy.

Například pokud používáte C, kód ve standardu ANSI C. Použijte standardní funkci knihovny C spíše než ekvivalentní funkci specifickou pro platformu, a to i v případě, že funkce specifická pro platformu je rychlejší nebo efektivnější. Výjimkou je, když je účinnost v kódu prvořadá, kdy byste měli kódovat pro obě situace pomocí #ifdef. Příklad:

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

```
    generic code
#endif
```

Typy front

Chcete vytvořit frontu pokaždé, když ji potřebujete, nebo chcete použít fronty, které již byly nastaveny? Chcete odstranit frontu, když ji dokončíte, nebo ji budete znovu používat? Chcete použít alias fronty pro nezávislost aplikace? Informace o podporovaných typech front naleznete v tématu Fronty.

z/OS Použití sdílených front, skupin sdílení front a klastrů skupin sdílení front (pouze IBM MQ for z/OS)

Možná budete chtít využít výhod zvýšené dostupnosti, rozšiřitelnosti a vyrovnávání pracovní zátěže, které jsou možné při použití sdílených front se skupinami sdílení front. Další informace viz Sdílené fronty a skupiny sdílení front .

Můžete také odhadnout průměrné a maximální toky zpráv a zvážit použití klastrů skupin sdílení front k rozložení pracovní zátěže. Další informace viz Sdílené fronty a skupiny sdílení front .

Použití klastrů správců front

Možná budete chtít využít výhod zjednodušené administrace systému a zvýšené dostupnosti, rozšiřitelnosti a vyrovnávání pracovní zátěže, které jsou možné při použití klastrů.

Typy zpráv

Možná budete chtít použít datagramy pro jednoduché zprávy, ale požadovat zprávy (pro které očekáváte odpovědi) pro jiné situace. Některým zprávám můžete přiřadit různé priority. Další informace o návrhu zpráv viz “Návrhové techniky pro zprávy” na stránce 55.

Použití publikování/odběru nebo dvoubodového systému zpráv

Pomocí systému zpráv publikování/odběru odesílající aplikace odešle informace, které chce sdílet ve zprávě IBM MQ , do standardního místa určení spravovaného produktem IBM MQ publish? subscribe, a umožní produktu IBM MQ zpracovat distribuci těchto informací. Cílová aplikace nemusí vědět nic o zdroji informací, které obdrží, pouze zaregistruje zájem o jedno nebo více témat a obdrží tuto informaci, až bude k dispozici. Další informace o systému zpráv publikování/odběru viz systém zpráv publikování/odběru.

Pomocí systému zpráv typu point-to-point odesílající aplikace odešle zprávu do určité fronty, odkud ví, že ji přijímající aplikace načte. Přijímající aplikace získává zprávy ze specifické fronty a pracuje s jejich obsahem. Aplikace bude často fungovat jako odesílatel i příjemce, odešle dotaz jiné aplikaci a obdrží odpověď.

Řízení programů IBM MQ

Můžete chtít spustit některé programy automaticky nebo nechat programy čekat na doručení určité zprávy do fronty (pomocí funkce IBM MQ *spouštění* , viz “Spuštění aplikací IBM MQ pomocí spouštěčů” na stránce 836). Alternativně můžete chtít spustit jinou instanci aplikace, když zprávy ve frontě nejsou dostatečně rychle zpracovány (pomocí funkce IBM MQ *události přípravy nástrojů* , jak je popsáno v tématu Události přípravy nástrojů).

Spuštění aplikace v klientu IBM MQ

Úplné rozhraní MQI je podporováno v prostředí klienta a téměř všechny aplikace IBM MQ napsané v procedurálním jazyce lze znovu propojit a spustit na serveru IBM MQ MQI client. Propojte aplikaci v produktu IBM MQ MQI client s knihovnou MQIC a nikoli s knihovnou MQI. z/OS Získání (signál) na systému z/OS není podporováno.

Poznámka: Aplikace spuštěná v klientu IBM MQ se může souběžně připojit k více než jednomu správci front nebo může použít název správce front s hvězdičkou (*) ve volání MQCONN nebo MQCONNX. Změňte aplikaci, chcete-li odkazovat na knihovny správce front namísto na knihovny klienta, protože tato funkce nebude k dispozici.

Další informace viz “Spuštění aplikací v prostředí IBM MQ MQI client” na stránce 889.

Výkon aplikace

Rozhodování o návrhu může mít dopad na výkon vaší aplikace, pokud jde o návrhy na zvýšení výkonu aplikací IBM MQ , viz “Aspekty návrhu aplikace a výkonu” na stránce 57 IBM i a “Aspekty návrhu a výkonu pro aplikace IBM i” na stránce 60 .

Pokročilé IBM MQ techniky

Pro pokročilejší aplikace můžete použít některé pokročilé techniky IBM MQ , jako např. korelování odpovědí a generování a odesílání informací o kontextu IBM MQ . Další informace viz “Návrhové techniky pro pokročilé aplikace” na stránce 58.

Zabezpečení dat a zachování jejich integrity

Informace o kontextu předané se zprávou můžete použít k otestování, zda byla zpráva odeslána z přijatelného zdroje. Prostředky synchronizace poskytované produktem IBM MQ nebo vaším operačním systémem můžete použít k zajištění toho, aby vaše data zůstala konzistentní s ostatními prostředky (další podrobnosti viz “Potvrzení a zálohování jednotek práce” na stránce 825). K zajištění doručení důležitých zpráv můžete použít funkci *persistence* pro zprávy IBM MQ .

Testování aplikací IBM MQ

Prostředí pro vývoj aplikací pro programy IBM MQ se nijak neliší od prostředí pro jiné aplikace, takže můžete použít stejné vývojové nástroje i prostředky trasování IBM MQ .

z/OS Při testování aplikací CICS pomocí produktu IBM MQ for z/OSmůžete použít nástroj CICS Execution Diagnostic Facility (CEDF). CEDF zachycuje vstup a výstup každého volání MQI a také volání všech služeb CICS . V prostředí CICS můžete také napsat uživatelský program pro přechod rozhraní API, který poskytne diagnostické informace před a po každém volání MQI. Informace o tom, jak to provést, viz “Používání a psaní aplikací na systému IBM MQ for z/OS” na stránce 858.

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

Zpracování výjimek a chyb

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

Související pojmy

IBM MQ Technický přehled

“Aspekty návrhu a výkonu pro aplikace z/OS” na stránce 62
Návrh aplikace je jedním z nejdůležitějších faktorů ovlivňujících výkon. Toto téma slouží k pochopení některých faktorů návrhu, které se podílejí na výkonu.

“Vyvíjení aplikací pro IBM MQ” na stránce 5
Můžete vyvíjet aplikace pro odesílání a příjem zpráv a pro správu správců front a souvisejících prostředků. Produkt IBM MQ podporuje aplikace napsané v mnoha různých jazycích a rámcích.

“Koncepty vývoje aplikací” na stránce 7
K psaní aplikací IBM MQ můžete použít výběr procedurálních nebo objektově orientovaných jazyků. Než začnete navrhovat a psát aplikace IBM MQ , seznamte se se základními koncepty produktu IBM MQ .

“Psaní procedurální žádosti o zařazení do fronty” na stránce 699
Prostřednictvím těchto informací můžete získat informace o psaní aplikací řazení do front, o připojování a odpojování od správce front, o publikování/odběru a o otevírání a zavírání objektů.

“Psaní procedurálních aplikací klienta” na stránce 882
Co potřebujete vědět, abyste mohli psát klientské aplikace na systému IBM MQ pomocí procedurálního jazyka.

“Vývoj aplikací .NET” na stránce 539
IBM MQ classes for .NET Povolit aplikacím .NET připojit se k produktu IBM MQ jako k serveru IBM MQ MQI client nebo se připojit přímo k serveru IBM MQ .

“Vývoj aplikací C++” na stránce 512
Produkt IBM MQ poskytuje třídy C++ ekvivalentní objektům IBM MQ a některé další třídy ekvivalentní datovým typům pole. Poskytuje řadu funkcí, které nejsou k dispozici prostřednictvím rozhraní MQI.

“Použití IBM MQ classes for JMS/Jakarta Messaging” na stránce 78
IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging jsou poskytovatelé systému zpráv Java dodávaní s produktem IBM MQ. Kromě implementace rozhraní definovaných ve specifikacích JMS

a Jakarta Messaging tito poskytovatelé systému zpráv přidávají do rozhraní API systému zpráv Java dvě sady rozšíření.

"Použití produktu IBM MQ classes for Java" na stránce 335
Použijte IBM MQ v prostředí Java . IBM MQ classes for Java Povolit aplikaci Java připojit se k produktu IBM MQ jako klient IBM MQ nebo se připojit přímo ke správci front IBM MQ .

Určení názvu aplikace v podporovaných programovacích jazycích

Před produktem IBM MQ 9.2.0již můžete zadat název aplikace v klientských aplikacích typu Java nebo JMS . Od produktu IBM MQ 9.2.0 je tato funkce rozšířena na další programovací jazyky v systému IBM MQ for Multiplatforms.

Jak se používá název aplikace

Název aplikace je výstupem z:

- Příkaz runmqsc DISPLAY CONN APPLTAG
- Příkaz runmqsc DISPLAY QSTATUS TYPE (HANDLE) APPLTAG
- Příkaz runmqsc DISPLAY CHSTATUS RAPPLTAG
- MQMD.PutApplName
- Trasování aktivity aplikace

Název aplikace se také používá při konfiguraci trasování aktivity aplikace. Výchozí název aplikace pro jiné aplikace nežJava je zkrácený název spustitelného souboru s výjimkou Windows a IBM i.

Windows V systému Windowsje výchozím názvem úplný název spustitelného souboru, zkrácený na 28 znaků vlevo.

IBM i V systému IBM ije výchozím názvem název úlohy.

V případě aplikací Java se jedná o název třídy s předponou s názvem balíku oříznutým vlevo na 28 znaků.

Další informace viz PutAppl.

V systému IBM MQ 9.2.0mohou aplikace v systému IBM MQ for Multiplatforms nastavovat názvy aplikací buď administrativně, nebo pomocí různých programovacích metod. To umožňuje aplikacím poskytovat smysluplnější název nezávislý na platformě při konfiguraci trasování aktivity aplikace nebo při výstupu z různých příkazů **runmqsc** .

V produktu IBM MQ 9.2.0můžete znovu vyvážit aplikace v rámci jednotného klastru. K dosažení tohoto cíle se používají smysluplné názvy aplikací.

Podporované znaky

Další informace o způsobu zadání názvu aplikace naleznete v části "Doporučené znaky názvu aplikace" na stránce 51 .

Programovací jazyky

Další informace o tom, jak mohou aplikace, které se interpretují do knihoven IBM MQ v jazyce C, a dalších programovacích jazycích, poskytnout název aplikace, naleznete v části "Připojení programovacího jazyka" na stránce 53 .

Spravované aplikace .NET

Informace o tom, jak mohou spravované aplikace .NET poskytovat název aplikace, naleznete v tématu "Spravované aplikace .NET" na stránce 54 .

Aplikace XMS

Informace o tom, jak mohou aplikace XMS poskytovat název aplikace, naleznete v části “Aplikace rozhraní XMS” na stránce 54 .

Aplikace vazeb Java a JMS

ALW

Informace o tom, jak mohou aplikace Java a JMS poskytnout název aplikace, naleznete v části “Aplikace vazeb Java a JMS” na stránce 55 .

Související pojmy

Trasování aktivity aplikace

O jednotných klastrech

Související odkazy

MQCNO

MQCNO na IBM i

Použití názvu aplikace v podporovaných programovacích jazycích

Pomocí těchto informací se dozvíte, jak je název aplikace vybrán v různých jazycích, které produkt IBM MQ podporuje.

Doporučené znaky názvu aplikace

Názvy aplikací musí být ve znakové sadě dané atributem **CodedCharSetId** pole správce front. Další informace o tomto atributu naleznete v tématu Atributy pro správce front.

Pokud je však aplikace spuštěna jako IBM MQ MQI client, název aplikace musí být ve znakové sadě a kódování klienta.

Chcete-li zajistit plynulý přechod názvu aplikace mezi správci front a povolit monitorování prostředků aplikace prostřednictvím témat monitorování prostředků, měly by názvy aplikací obsahovat pouze jednobajtové tisknutelné znaky.

Notes:

- Také byste se měli vyhnout použití dopředného lomítka a znaků ampersand v názvech aplikací.
- V 9.3.0 Měli byste se vyhnout použití znaku ampersand v názvech aplikací. Metriky STATAPP tématu systému pro názvy aplikací obsahující znak ampersand nebudou vytvořeny.

Tím se název omezí na:

- Alfanumerické znaky: `A-Z, a-za 0-9`

  Poznámka: V názvech aplikací byste neměli používat malá písmena `a-z` v systémech používajících EBCDIC Katakana.
- Znak mezery
- Tisknutelné znaky, které jsou invariantní v EBCDIC: `+ < = > % * ' ( ) , _ - . : ; ?`
- V 9.3.0 Znak/. Při přihlášení k odběru metrik systémového tématu trasování aktivity nebo STATAPP pro aplikaci, jejíž název obsahuje dopředné lomítko, musíte nahradit všechny dopředné lomítko znakem ampersand. Chcete-li například přijmout metriky STATAPP pro aplikaci s názvem "DEPT1/APPS/STOCKQUOTE", musíte se přihlásit k odběru řetězce tématu "$SYS/MQ/INFO/QMGR/QMBASIC/Monitor/STATAPP/DEPT1&APPS&STOCKQUOTE/INSTANCE". Ukázkové aplikace amqsact a amqsrua při vytváření svých odběrů automaticky převedou dopředné lomítko na znak ampersand.

Jak nastavit znaky

Následující tabulka shrnuje prostředky, podle kterých je název aplikace vybrán v různých jazycích, které produkt IBM MQ podporuje. Způsob, kterým je název vybrán, je v pořadí podle priority, nejprve nejvyšší.

	Vazby C a klient	**Vazby Java a klient**	**Vazby JMS a klient**	**Spravovaný klient .NET**	**Nespravované vazby .NET a klient**	**Spravovaný klient XMS**	**Nespravované . Vazby a klient XMS**
Potlačení vlastnosti připojení		Java potlačení vlastnosti připojení		.Potlačení vlastnosti připojení NET	.Potlačení vlastnosti připojení NET		
Přepsaná vlastnost		Java přepsaná vlastnost		.Přepsaná vlastnost NET	.Přepsaná vlastnost NET		
Prostředí MQEnvironment		Java Prostředí MQEnvironment		.Prostředí NET MQEnvironment	.Prostředí NET MQEnvironment		
Vlastnost továrny připojení			Vlastnost továrny připojení			Vlastnost továrny připojení	Vlastnost továrny připojení
JMSAdmin			JMSAdmin			JMSAdmin	JMSAdmin
MQCNO	Volby připojení						
Proměnná prostředí	Proměnné prostředí				Proměnné prostředí		Proměnné prostředí
mqclient.ini (Platí pouze pro klientská připojení)	Připojení klienta				Připojení klienta		Připojení klienta
Java Název třídy		Java název třídy	Java název třídy				
Výchozí název	Výchozí název			.Výchozí název NET	.Výchozí název NET	.Výchozí název NET	.Výchozí název NET

Poznámka: Vazby C a sloupec klienta se vztahují i na následující programovací jazyky:

- COBOL
- Sestavovací modul
- Visual Basic
- RPG, se

Připojení programovacího jazyka

Aplikace, které se interpretují do knihoven IBM MQ v jazyce C a dalších programovacích jazycích, mohou poskytnout název aplikace následujícími způsoby.

Metody připojení jsou uvedeny v pořadí podle priority, počínaje nejvyšším.

Multi **Volby připojení**

- ALW MQCNO

  Poznámka: z/OS Při připojování ke správci front IBM MQ for z/OS můžete nastavit název aplikace pouze pomocí připojení v režimu klienta nebo pomocí aplikací IBM MQ classes for JMS či IBM MQ classes for Java .

- IBM i MQCNO v systému IBM i

ALW **Proměnné prostředí**

Pokud jste dosud nevybrali název aplikace, můžete k identifikaci připojení ke správci front použít proměnnou prostředí **MQAPPLNAME** . Příklad:

```
export MQAPPLNAME=ExampleApplName
```

Všimněte si, že se použije pouze prvních 28 znaků a tyto znaky nesmí být všechny mezery nebo hodnoty null.

Poznámka: Tento atribut platí pouze pro podporovaná programovací jazyky, nespravovaná připojení produktu .NETa nespravovaná připojení produktu XMS .

ALW **Konfigurační soubor klienta**

Pokud jste dosud nevybrali název aplikace a připojení je připojení klienta, můžete v konfiguračním souboru klienta (například `mqclient.ini`) určit následující informace pro identifikaci připojení ke správci front.

```
Connection:
    ApplName=ExampleApplName
```

Notes:

1. Použije se pouze prvních 28 znaků a tyto znaky nesmí být všechny mezery nebo hodnoty null.
2. Atribut se vztahuje pouze na připojení klienta v podporovaných programovacích jazycích, nespravovaných .NETa nespravovaných XMS připojeních.

Další informace viz IBM MQ MQI client konfigurační soubor, `mqclient.ini`.

Výchozí název

Pokud jste dosud nevybrali název aplikace, bude nadále používán výchozí název, který obsahuje tolik cesty a spustitelného názvu, kolik zobrazuje operační systém. Další informace viz PutAppl.

Spravované aplikace .NET

Spravované aplikace .NET mohou poskytnout název aplikace následujícími způsoby.

Metody připojení jsou uvedeny v pořadí podle priority, počínaje nejvyšším.

potlačení vlastnosti připojení

Můžete poskytnout soubor potlačení podrobností připojení pro aplikace následujícím způsobem:

```
<appSettings>
     <add key="overrideConnectionDetails" value="true" />
     <add key="overrideConnectionDetailsFile" value="<location>" />
</appSettings>
```

Soubor určený parametrem `overrideConnectionDetailsFile` obsahuje seznam vlastností s předponou `mqj`. Aplikace musí definovat vlastnost **`mqj.APPNAME`** , kde hodnota vlastnosti **`mqj.APPNAME`** určuje název použitý k identifikaci připojení ke správci front.

Použije se pouze prvních 28 znaků názvu. Příklad:

```
mqj.APPNAME=ExampleApplName
```

Přepsaná vlastnost

Konstanta **`MQC.APPNAME_PROPERTY`** byla definována s hodnotou *APPNAME*. Nyní můžete předat tuto vlastnost konstruktoru **`MQQueueManager`** s použitím prvních 28 znaků pouze názvu. Příklad:

```
Hashtable properties = new Hashtable();
properties.Add( MQC.APPNAME_PROPERTY, "ExampleApplName" );
MQQueueManager qMgr = new MQQueueManager("qmgrname", properties);
```

Další informace viz téma “Spravované a nespravované operace v produktu .NET” na stránce 634.

MQEnvironment

Vlastnost *AppName* se přidá do třídy **`MQEnvironment`**a použije se pouze prvních 28 znaků. Příklad:

```
MQEnvironment.AppName = "ExampleApplName";
```

Výchozí název

Pokud jste nezadali název aplikace žádným z prostředků popsaných v předchozím textu, bude název aplikace automaticky nastaven jako název spustitelného souboru (a stejně tak jako cesta, která se vejde).

Aplikace rozhraní XMS

Metody připojení jsou uvedeny v pořadí podle priority, počínaje nejvyšším.

Vlastnost továrny připojení

Aplikace XMS mohou poskytnout název aplikace v továrně připojení pomocí vlastnosti **`XMSC.WMQ_APPLICATIONNAME`** ("*XMSC_WMQ_APPNAME*") podobným způsobem jako JMS. Můžete zadat až 28 znaků.

Další informace naleznete v tématech “XMS .NET vytváření spravovaných objektů” na stránce 642 a “Vlastnosti zprávy XMS” na stránce 649.

JMSAdmin

V administrativních nástrojích je vlastnost zkráceně označována jako "**`APPLICATIONNAME`**" nebo "**`APPNAME`**".

Aplikace vazeb Java a JMS

Metody připojení jsou uvedeny v pořadí podle priority, počínaje nejvyšším.

ALW Aplikace klienta Java a JMS již mohou zadat název aplikace, který byl v systému IBM MQ for Multiplatforms rozšířen na aplikace vazeb pomocí pole MQCNO **ApplName** .

potlačení vlastnosti připojení

Vlastnost **Application name** byla přidána do seznamu vlastností připojení, které můžete přepsat. Další informace naleznete v tématu Použití IBM MQ potlačení vlastnosti připojení.

Upozornění: Vlastnosti připojení a způsob použití souboru přepsání vlastností připojení jsou stejné pro IBM MQ classes for Java i .NET.

Přepsaná vlastnost

Konstanta **MQC.APPNAME_PROPERTY** byla definována s hodnotou *APPNAME*. Nyní můžete předat tuto vlastnost konstruktoru **MQQueueManager** s použitím prvních 28 znaků pouze názvu. Další informace naleznete v tématu Použití potlačení vlastnosti připojení v produktu IBM MQ classes for Java.

MQEnvironment

Vlastnost *AppName* se přidá do třídy **MQEnvironment** a použije se pouze prvních 28 znaků.

Další informace viz téma “Nastavení prostředí IBM MQ pro IBM MQ classes for Java” na stránce 362.

Java název třídy

Pokud jste nezadali název aplikace žádným ze způsobů uvedených v předchozím textu, je název aplikace odvozen od názvu hlavní třídy.

Další informace viz téma “Nastavení prostředí IBM MQ pro IBM MQ classes for Java” na stránce 362.

Upozornění: IBM i V systému IBM i není možné zadat dotaz na název hlavní třídy, takže se místo toho použije IBM MQ client for Java .

Související pojmy

“Nastavení prostředí IBM MQ pro IBM MQ classes for Java” na stránce 362
Aby se aplikace mohla připojit ke správci front v režimu klienta, musí zadat název kanálu, název hostitele a číslo portu.

Související odkazy

MQCNO
MQCNO na IBM i

Návrhové techniky pro zprávy

Aspekty, které vám pomohou při návrhu zpráv, včetně aspektů pro selektory a vlastnosti zpráv.

Co je třeba zvážit ve fázi návrhu

Zprávu vytvoříte při použití volání MQI pro vložení zprávy do fronty. Jako vstup pro volání zadáte některé řídicí informace do *deskriptoru zprávy* (MQMD) a data, která chcete odeslat do jiného programu. Ale ve fázi návrhu musíte zvážit následující, protože ovlivňují způsob, jakým vytváříte zprávy:

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

Chcete, aby byly zprávy požadavku a odpovědi synchronní? To znamená, že nastavíte časový limit pro odpověď, aby odpověděla na váš požadavek, a pokud odpověď v tomto období neobdržíte, bude se s ní zacházet jako s chybou.

Nebo byste raději pracovali asynchronně, aby vaše procesy nemusely záviset na výskytu specifických událostí, jako jsou běžné časové signály?

Další úvaha je, zda máte všechny své zprávy uvnitř jednotky práce.

Přiřazení různých priorit zprávám

Ke každé zprávě můžete přiřadit hodnotu priority a definovat frontu tak, aby udržovala své zprávy v pořadí podle jejich priority. Pokud tak učiníte, když jiný program načte zprávu z fronty, vždy obdrží zprávu s nejvyšší prioritou. Pokud fronta neudržuje své zprávy v pořadí podle priority, program, který načítá zprávy z fronty, je načte v pořadí, ve kterém byly přidány do fronty.

Programy mohou také vybrat zprávu pomocí identifikátoru, který správce front přiřadil při vložení zprávy do fronty. Případně můžete generovat vlastní identifikátory pro každou z vašich zpráv.

Vliv restartování správce front na zprávy

Správce front při restartování zachová všechny trvalé zprávy a v případě potřeby je obnoví ze souborů protokolu IBM MQ . Dočasné zprávy a dočasné dynamické fronty nejsou zachovány. Všechny zprávy, které nechcete vyřadit, musí být při vytváření definovány jako trvalé. Při zápisu aplikace pro systém IBM MQ for Windows nebo IBM MQ na systémech AIX and Linux se ujistěte, že víte, jak byl váš systém nastaven s ohledem na alokaci souborů protokolu, abyste snížili riziko návrhu aplikace, která bude spuštěna s limity souborů protokolu.

z/OS Vzhledem k tomu, že zprávy ve sdílených frontách (k dispozici pouze v systému IBM MQ for z/OS) jsou uloženy v prostředku Coupling Facility (CF), přechodné zprávy jsou zachovány v rámci restartů správce front, dokud je prostředek CF k dispozici. Pokud prostředek CF selže, přechodné zprávy budou ztraceny.

Poskytování informací o sobě příjemci zpráv

Správce front obvykle nastaví ID uživatele, ale vhodně autorizované aplikace mohou také nastavit toto pole, takže můžete zahrnout vlastní ID uživatele a další informace, které může přijímající program použít pro účely evidence nebo zabezpečení.

Množství přijímacích front

Multi Pokud může být nutné zprávu vložit do několika front, můžete ji publikovat v tématu nebo v distribučním seznamu.

z/OS Pokud může být nutné vložit zprávu do několika front, můžete ji publikovat v tématu.

Selektory a vlastnosti zpráv

Ke zprávám mohou být vedle hlavního informačního obsahu zprávy přidružena metadata. Tyto vlastnosti zprávy mohou být užitečné při zadávání dalších dat.

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

- Vlastnosti nepodléhají ochraně Advanced Message Security (AMS). Chcete-li k ochraně dat použít AMS, vložte jej do informačního obsahu a ne do vlastností zprávy.
- Vlastnosti lze použít k provedení výběru zpráv.

Je důležité si uvědomit, že použití selektorů porušuje standardní konvenci zpráv prvního v prvním. Vzhledem k tomu, že je správce front optimalizován pro tuto pracovní zátěž, nedoporučuje se z důvodu výkonu poskytovat složité selektory. Správce front neukládá indexy vlastností zprávy, proto hledání zprávy musí být lineární vyhledávání. Hlubší fronta, složitější selektor a nižší pravděpodobnost, že selektor odpovídající zprávě nepříznivě ovlivní výkon.

Je-li vyžadován komplexní výběr, doporučuje se filtrovat zprávy pomocí libovolné aplikace nebo stroje pro zpracování, například IBM Integration Bus, do různých míst určení. Alternativně může být užitečné použití hierarchie témat.

Poznámka: IBM MQ classes for Java nepodporují použití selektorů, pokud si přejete použít selektory, měly by být provedeny prostřednictvím rozhraní API JMS .

Aspekty návrhu aplikace a výkonu

Existuje řada způsobů, jak může špatný návrh programu ovlivnit výkon. Ty mohou být obtížně detekovatelné, protože se může zdát, že program provádí dobře sám, ale ovlivňuje výkon jiných úloh. V tomto tématu je vysvětleno několik problémů specifických pro volání programů IBM MQ .

Zde je několik nápadů, které vám pomohou navrhnout efektivní aplikace:

- Navrhněte svou aplikaci tak, aby zpracování probíhalo souběžně s dobou přemýšlení uživatele:
  - Zobrazte panel a umožněte uživateli začít psát, zatímco se aplikace stále inicializuje.
  - Získejte data, která potřebujete paralelně, z různých serverů.
- Udržujte připojení a fronty otevřené, pokud je budete opakovaně používat namísto opakovaného otevírání a zavírání, připojování a odpojování.
- Serverová aplikace, která vkládá pouze jednu zprávu, by však měla používat MQPUT1.
- Správci front jsou optimalizováni pro zprávy o velikosti od 4 kB do 100 kB. Velmi velké zprávy jsou neefektivní; je pravděpodobně lepší odeslat 100 zpráv o velikosti 1 MB na každou než jednu zprávu o velikosti 100 MB. 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 jednotky práce, aby mohly být potvrzeny nebo vráceny zpět současně.
- Pro zprávy, které nemusí být obnovitelné, použijte dočasnou volbu.
- Potřebujete-li odeslat zprávu do několika cílových front, zvažte použití rozdělovníku.

Vliv délky zprávy

Množství dat ve zprávě může ovlivnit výkon aplikace, která zprávu zpracovává. Chcete-li dosáhnout nejlepšího výkonu z vaší aplikace, odešlete ve zprávě pouze nezbytná data. Například v požadavku na debetní bankovní účet jsou jedinými informacemi, které mohou být nutné předat z klienta do serverové aplikace, číslo účtu a částka debetu.

Účinek perzistence zpráv

Trvalé zprávy jsou obvykle protokolovány. Protokolování zpráv snižuje výkon aplikace, proto používejte trvalé zprávy pouze pro základní data. Pokud mohou být data ve zprávě vyřazena v případě, že se správce front zastaví nebo selže, použijte dočasnou zprávu.

z/OS Operace MQPUT a MQGET pro trvalé zprávy budou blokovány v případě, že pro záznam operací není k dispozici dostatek prostoru pro žurnál zotavení. Taková podmínka je v protokolu úlohy správce front označena zprávami CSQJ110E a CSQJ111A. Zajistěte, aby byly zavedeny procesy monitorování, aby bylo možné tyto podmínky spravovat a vyhnout se jim.

Hledání konkrétní zprávy

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

Fronty, které obsahují zprávy různých délek

Pokud vaše aplikace nemůže používat zprávy s pevnou délkou, zvětšete a zmenšete vyrovnávací paměti dynamicky tak, aby vyhovovaly typické velikosti zprávy. Pokud aplikace vydá volání MQGET, které selže, protože vyrovnávací paměť je příliš malá, vrátí se velikost dat zprávy. Přidejte do aplikace kód, aby byla vyrovnávací paměť odpovídajícím způsobem změněna a bylo znovu vydáno volání MQGET.

Poznámka: Pokud atribut **MaxMsgLength** nenastavíte explicitně, výchozí hodnota je 4 MB, což může být velmi neefektivní, pokud se používá k ovlivnění velikosti vyrovnávací paměti aplikace.

Frekvence synchronizačních bodů

Programy, které vydávají velký počet volání MQPUT nebo MQGET v synchronizačním bodu bez jejich potvrzení, mohou způsobit problémy s výkonem. Ovlivněné fronty se mohou zaplnit zprávami, které jsou momentálně nepřístupné, zatímco jiné úlohy mohou čekat na získání těchto zpráv. To má důsledky z hlediska úložiště a z hlediska podprocesů, které jsou svázány s úlohami, které se pokoušejí získat zprávy.

Použití volání MQPUT1

Volání MQPUT1 použijte pouze v případě, že máte jednu zprávu, kterou chcete vložit do fronty. Chcete-li vložit více než jednu zprávu, použijte volání MQOPEN následované řadou volání MQPUT a jedním voláním MQCLOSE.

Počet používaných podprocesů

Windows V případě systému IBM MQ for Windowsmůže aplikace vyžadovat velký počet podprocesů. Každému procesu správce front je přidělen maximální povolený počet podprocesů aplikace.

Aplikace mohou používat příliš mnoho podprocesů. Zvažte, zda aplikace bere tuto možnost v úvahu a zda provádí akce k zastavení nebo nahlášení tohoto typu výskytu.

Vložit trvalé zprávy do synchronizačního bodu

Trvalé zprávy by měly být vloženy a dostaly se pod synchronizační bod. Je to proto, že při získávání trvalé zprávy mimo synchronizační bod, pokud se příkaz get nezdaří, není možné, aby aplikace věděla, zda byla zpráva získána z fronty či nikoli, a zda, pokud byla zpráva získána, byla také ztracena. Pokud při získávání trvalých zpráv v synchronizačním bodu dojde k selhání, transakce bude odvolána a trvalá zpráva nebude ztracena, protože je stále ve frontě.

Podobně při vkládání trvalých zpráv je umístěte do synchronizačního bodu. Dalším důvodem pro vložení a získání trvalých zpráv do synchronizačního bodu je skutečnost, že kód trvalé zprávy v produktu IBM MQ je silně optimalizován pro synchronizační bod. Takže vkládání a získávání trvalých zpráv v synchronizačním bodu je rychlejší než vkládání a získávání trvalých zpráv mimo synchronizační bod.

Pokud vaše aplikace vloží trvalé zprávy mimo synchronizační bod, správce front zkontroluje, zda může vytvořit implicitní synchronizační bod jménem aplikace. Pokud tak správce front může učinit, zahrne vložení do tohoto synchronizačního bodu a automaticky jej potvrdí. Podrobnější popis viz “Implicitní synchronizační bod na multiplatformách” na stránce 833 .

Je však rychlejší vkládat a získávat dočasné zprávy mimo synchronizační bod, protože dočasný kód v produktu IBM MQ je optimalizován pro to, aby byl mimo synchronizační bod. Vkládání a získávání trvalých zpráv probíhá rychlostí disku, protože trvalá zpráva je trvale uložena na disku. Vkládání a získávání dočasných zpráv však probíhá rychlostí procesoru, protože není zapojen žádný zápis na disk, a to ani při použití synchronizačního bodu.

Pokud aplikace získává zprávy a předem neví, zda jsou trvalé, či nikoli, lze použít volbu GMO MQGMO_SYNCPOINT_IF_PERSISTENT.

Návrhové techniky pro pokročilé aplikace

Při návrhu pokročilejších aplikací můžete zvážit některé techniky, například čekání na zprávy, korelování odpovědí, nastavení a používání kontextových informací, automatické spouštění aplikací, generování sestav a odebrání afinit zpráv při použití klastrování.

V případě jednoduché aplikace IBM MQ se musíte rozhodnout, které objekty IBM MQ se mají ve vaší aplikaci použít, a jaké typy zpráv chcete použít. Pro pokročilejší aplikaci můžete použít některé z technik uvedených v následujících sekcích.

Čekání na zprávy

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

- Čekání na doručení zprávy nebo na vypršení určeného časového intervalu (viz “Čekání na zprávy” na stránce 771).
- z/OS Pouze v systému IBM MQ for z/OS nastavení signálu tak, aby byl program informován při doručení zprávy. Další informace viz téma “signalizace” na stránce 772.
- Zavedení uživatelské procedury zpětného volání, která má být řízena při doručení zprávy; viz “Asynchronní spotřeba zpráv IBM MQ” na stránce 40.
- Provádění pravidelných volání ve frontě za účelem zjištění, zda zpráva dorazila (*systém výzev*). To se obvykle nedoporučuje, protože to může mít dopad na výkon.

Související odpovědi

V aplikacích IBM MQ , když program obdrží zprávu, která jej požaduje, aby vykonal nějakou práci, program obvykle odešle žadateli jednu nebo více zpráv s odpovědí.

Chcete-li žadateli pomoci přidružit tyto odpovědi k jeho původnímu požadavku, může aplikace nastavit pole *identifikátor korelace* v deskriptoru každé zprávy. Programy pak zkopírují identifikátor zprávy požadavku do pole identifikátoru korelace jejich zpráv odpovědi.

Nastavení a použití informací o kontextu

Informace o kontextu se používají k přidružení zpráv k uživateli, který je vygeneroval, a k identifikaci aplikace, která zprávu vygenerovala. Tyto informace jsou užitečné pro zabezpečení, účetnictví, auditování a určování problémů.

Při vytváření zprávy můžete zadat volbu, která vyžaduje, aby správce front přidružil k vaší zprávě výchozí informace o kontextu.

Další informace o použití a nastavení informací o kontextu viz “kontext zprávy” na stránce 45.

Automatické spuštění programů IBM MQ

IBM MQ *spouštění* použijte k automatickému spuštění programu, když zprávy dorazí do fronty.

Můžete nastavit podmínky spouštěče ve frontě tak, aby program spustil zpracování této fronty:

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

Další informace o spouštění viz “Spuštění aplikací IBM MQ pomocí spouštěčů” na stránce 836. Spuštění je jen jeden způsob, jak spustit program automaticky. Můžete například spustit program automaticky na časovači pomocí jiných zařízení nežIBM MQ .

Multi V systému Multiplatformsmůže produkt IBM MQ definovat objekty služeb ke spuštění programů IBM MQ při spuštění správce front. Viz Objekty služeb.

Generování sestav IBM MQ

V rámci aplikace si můžete vyžádat následující sestavy:

- Sestavy výjimek
- Sestavy vypršení platnosti
- Sestavy potvrzení při příjezdu (COA)
- Sestavy potvrzení při doručení (COD)
- Pozitivní zprávy o oznámení akce (PAN)

- Negativní zprávy o oznámení akce (NAN)

Ty jsou popsány v části "Zprávy sestavy" na stránce 20.

Klastry a afinity zpráv

Než začnete používat klastry s více definicemi pro stejnou frontu, prozkoumejte aplikace, abyste zjistili, zda existují nějaké, které vyžadují výměnu souvisejících zpráv.

V rámci klastru lze zprávu směrovat na libovolného správce front, který je hostitelem instance příslušné fronty. Proto může být narušena logika aplikací s afinitou zpráv.

Můžete mít například dvě aplikace, které spoléhají na řadu zpráv, které mezi nimi proudí ve formě otázek a odpovědí. Může být důležité, aby všechny otázky byly odeslány stejnému správci front a všechny odpovědi byly odeslány zpět druhému správci front. V této situaci je důležité, aby rutina správy pracovní zátěže neodesílala zprávy žádnému správci front, který je právě hostitelem instance příslušné fronty.

Pokud je to možné, odstraňte afinity. Odebrání afinit zpráv zlepšuje dostupnost a rozšiřitelnost aplikací.

Další informace naleznete v tématu Zpracování afinit zpráv.

Aspekty návrhu a výkonu pro aplikace IBM i

Tyto informace vám pomohou porozumět tomu, jak může návrh aplikace, podprocesy a úložiště ovlivnit výkon.

Tyto informace jsou rozděleny do dvou sekcí:

- "Aspekty návrhu aplikace" na stránce 60
- "Specifické problémy s výkonem" na stránce 61

Aspekty návrhu aplikace

Existuje řada způsobů, jak může špatný návrh programu ovlivnit výkon. Tyto problémy mohou být obtížně detekovatelné, protože se může zdát, že program provádí dobře, a zároveň ovlivňuje výkon jiných úloh. Několik problémů specifických pro volání programů IBM MQ for IBM i je vysvětleno v následujících sekcích.

Další informace o návrhu aplikace viz "Aspekty návrhu pro aplikace IBM MQ" na stránce 47.

Vliv délky zprávy
Ačkoli produkt IBM MQ for IBM i umožňuje, aby zprávy pojaly až 100 MB dat, množství dat ve zprávě ovlivňuje výkon aplikace, která zprávu zpracovává. Chcete-li dosáhnout nejlepšího výkonu z vaší aplikace, odešlete pouze základní data ve zprávě; například v požadavku na debetní bankovní účet je jedinou informací, kterou může klient předat serverové aplikaci, číslo účtu a částka debetu.

Účinek perzistence zpráv
Trvalé zprávy jsou žurnálovány. Žurnálování zpráv snižuje výkon aplikace, proto používejte trvalé zprávy pouze pro základní data. Pokud mohou být data ve zprávě vyřazena v případě, že se správce front zastaví nebo selže, použijte dočasnou zprávu.

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

Fronty, které obsahují zprávy různých délek
Pokud mají zprávy ve frontě různé délky, může aplikace k určení velikosti zprávy použít volání MQGET s polem *BufferLength* nastaveným na nulu, takže i když volání selže, vrátí velikost dat zprávy. Aplikace pak může volání opakovat a určit identifikátor zprávy, kterou měřila při prvním volání, a vyrovnávací paměť správné velikosti. Pokud však existují další aplikace obsluhující stejnou frontu, můžete zjistit, že výkon aplikace je snížen, protože druhé volání MQGET stráví hledáním zprávy, kterou jiná aplikace načetla v době mezi dvěma voláními.

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

Frekvence synchronizačních bodů
Programy, které v synchronizačním bodu vyvolávají řadu volání MQPUT, aniž by je potvrzovaly, mohou způsobit problémy s výkonem. Ovlivněné fronty se mohou zaplnit zprávami, které jsou momentálně nepoužitelné, zatímco jiné úlohy mohou čekat na získání těchto zpráv. Tento problém má důsledky, pokud jde o úložiště, a pokud jde o podprocesy svázané s úlohami, které se pokoušejí získat zprávy.

Použití volání MQPUT1
Volání MQPUT1 použijte pouze v případě, že máte jednu zprávu, kterou chcete vložit do fronty. Chcete-li vložit více než jednu zprávu, použijte volání MQOPEN následované řadou volání MQPUT a jedním voláním MQCLOSE.

Počet používaných podprocesů
Aplikace může vyžadovat mnoho podprocesů. Každému procesu správce front je přidělen maximální povolený počet podprocesů. Pokud jsou některé aplikace problematické, může to být způsobeno jejich návrhem s použitím příliš mnoha podprocesů. Zvažte, zda aplikace bere tuto možnost v úvahu a zda provádí akce k zastavení nebo nahlášení tohoto typu výskytu. Maximální počet podprocesů, které produkt IBM i povoluje, je 4 095. Výchozí hodnota je však 64. Produkt IBM MQ zpřístupňuje až 63 podprocesů pro své procesy.

Specifické problémy s výkonem

Tento oddíl vysvětluje problémy s úložištěm a nedostatečný výkon.

Problémy s úložištěm
Pokud obdržíte systémovou zprávu `CPF0907. Serious storage condition may exist.`, je možné, že zaplňujete prostor přidružený ke správcům front IBM MQ for IBM i .

Běží vaše aplikace nebo IBM MQ for IBM i pomalu?
Pokud vaše aplikace běží pomalu, může to znamenat, že je ve smyčce, nebo že čeká na prostředek, který není k dispozici. Toto pomalé spuštění může být také způsobeno problémem s výkonem. Možná je to proto, že váš systém pracuje v blízkosti limitů své kapacity. Tento typ problému je pravděpodobně nejhorší v době nejvyššího zatížení systému, obvykle v dopoledních hodinách a v polovině odpoledne. (Pokud se vaše síť rozprostírá přes více než jedno časové pásmo, může se vám zdát, že se vám někdy v jiný čas vyskytne špičková zátěž systému.)

Pokud zjistíte, že snížení výkonu není závislé na zatížení systému, ale někdy se stane, když je systém lehce načten, je pravděpodobně na vině špatně navržený aplikační program. Tento problém se může projevit jako problém, který se vyskytuje pouze při přístupu k určitým frontám.

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

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

- Pokud váš systém reaguje na příkazy MQSC pomalu.
- Pokud se opakují zobrazení hloubky fronty, znamená to, že se fronta zpracovává pomalu pro aplikaci, se kterou byste očekávali velké množství aktivity fronty.
- Je trasování IBM MQ spuštěno?

Linux Aspekty návrhu pro aplikace Linux on Power Systems - Little Endian

Vzhledem k tomu, že produkt Linux on Power Systems - Little Endian podporuje pouze 64bitové aplikace, produkt IBM MQ nepodporuje 32bitové aplikace.

Související pojmy

“Aspekty návrhu pro aplikace IBM MQ” na stránce 47

Když jste se rozhodli, jak mohou vaše aplikace využívat platformy a prostředí, která máte k dispozici, musíte se rozhodnout, jak používat funkce nabízené produktem IBM MQ.

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

Návrh aplikace je jedním z nejdůležitějších faktorů ovlivňujících výkon. Toto téma slouží k pochopení některých faktorů návrhu, které se podílejí na výkonu.

Existuje řada způsobů, jak může špatný návrh programu ovlivnit výkon. Tyto problémy mohou být obtížně detekovatelné, protože se může zdát, že program provádí dobře, a zároveň ovlivňuje výkon jiných úloh. Několik problémů specifických pro programy, které volají rozhraní MQI, je ukázáno v následujících sekcích.

Další informace o návrhu aplikace viz “Aspekty návrhu pro aplikace IBM MQ” na stránce 47.

Vliv délky zprávy

Ačkoli produkt IBM MQ for z/OS umožňuje, aby zprávy pojaly až 100 MB dat, množství dat ve zprávě ovlivňuje výkon aplikace, která zprávu zpracovává. Chcete-li dosáhnout nejlepšího výkonu z vaší aplikace, odešlete ve zprávě pouze nezbytná data. Například v požadavku na debetní bankovní účet jsou jedinými informacemi, které mohou být potřeba předat z klienta do serverové aplikace, číslo účtu a částka na debetní účet.

Účinek perzistence zpráv

Trvalé zprávy jsou protokolovány. Protokolování zpráv snižuje výkon aplikace, proto používejte trvalé zprávy pouze pro základní data. Pokud mohou být data ve zprávě vyřazena v případě, že se správce front zastaví nebo selže, použijte dočasnou zprávu.

Data pro trvalé zprávy se zapisují do vyrovnávacích pamětí protokolu. Tyto vyrovnávací paměti se zapisují do datových sad protokolu, když:

- Dojde k potvrzení
- Zpráva je získána nebo vložena ze synchronizačního bodu
- WRTHRSH vyrovnávací paměti jsou vyplněny

Zpracování mnoha zpráv v jedné pracovní jednotce může způsobit méně vstupu/výstupu, než kdyby byly zprávy zpracovány pro každou pracovní jednotku, nebo mimo synchronizační bod.

Hledání konkrétní zprávy

Volání MQGET obvykle načte první zprávu z fronty. Používáte-li identifikátory zprávy a korelace (**MsgId** a **CorrelId**). v deskriptoru zprávy pro určení konkrétní zprávy správce front prohledává frontu, dokud tuto zprávu nenajde. Použití příkazu MQGET tímto způsobem ovlivňuje výkon aplikace, protože k nalezení konkrétní zprávy může být nutné, aby produkt IBM MQ skenoval celou frontu.

Pomocí atributu fronty **IndexType** můžete určit, že má správce front udržovat index, který lze použít ke zvýšení rychlosti operací MQGET ve frontě. Při údržbě indexu však dochází k malému snížení výkonu, takže jej generujte pouze v případě, že jej potřebujete použít. Můžete se rozhodnout sestavit index identifikátorů zpráv nebo identifikátorů korelace, nebo se můžete rozhodnout nesestavovat index pro fronty, kde jsou zprávy načítány sekvenčně. Pokuste se mít mnoho různých klíčových hodnot, ne loty se stejnou hodnotou. Například Balance1, Balance2a Balance3, ne tři se zůstatkem. Pro sdílené fronty musíte mít správnou hodnotu **IndexType**. Podrobnosti o atributu fronty **IndexType** viz IndexType.

Chcete-li se vyhnout ovlivnění času restartování správce front pomocí indexovaných front, použijte parametr QINDXBLD (NOWAIT) v makru CSQ6SYSP . To umožňuje dokončení restartu správce front bez čekání na dokončení sestavení indexu fronty.

Úplný popis atributu **IndexType** a dalších atributů objektu viz Atributy objektů.

Fronty, které obsahují zprávy různých délek

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 se příkaz get tímto způsobem nezdaří, vrácená délka dat je velikost nepřevedených dat zprávy. Zadáte-li parametr MQGMO_CONVERT pro volání MQGET a data se během převodu rozbalí, nemusí se přesto vejít do vyrovnávací paměti. V takovém případě je třeba dále zvětšit velikost vyrovnávací paměti.

Zadáte-li příkaz MQGET s nulovou délkou vyrovnávací paměti, vrátí velikost zprávy a aplikace pak může získat vyrovnávací paměť této velikosti a znovu zadat příkaz get. Máte-li více aplikací, které zpracovávají frontu, je možné, že jiná aplikace již zpracovala zprávu, když původní aplikace znovu vydala příkaz get. Máte-li příležitostně velké zprávy, možná budete muset získat velkou vyrovnávací paměť právě pro tyto zprávy a uvolnit ji po zpracování zprávy. To by mělo pomoci snížit problémy s virtuálním úložištěm, pokud mají všechny aplikace velké vyrovnávací paměti.

Pokud vaše aplikace nemůže používat zprávy s pevnou délkou, dalším řešením tohoto problému je použít volání MQINQ k nalezení maximální velikosti zpráv, které může fronta přijmout, a poté tuto hodnotu použít ve volání MQGET . Maximální velikost zpráv pro frontu je uložena v atributu **MaxMsgL** fronty. Tato metoda může používat velké množství úložiště, protože hodnota **MaxMsgL** může být až 100 MB, což je maximum povolené produktem IBM MQ for z/OS.

Poznámka: Po vložení velkých zpráv do fronty můžete snížit hodnotu parametru **MaxMsgL** . Můžete například vložit zprávu o velikosti 100 MB a poté nastavit **MaxMsgL** na 50 bajtů. To znamená, že je stále možné získat větší zprávy, než očekávala aplikace.

Frekvence synchronizovaných bodů

Programy, které vydávají mnoho volání MQPUT v rámci synchronizačního bodu bez jejich potvrzení, mohou způsobit problémy s výkonem. Ovlivněné fronty se mohou zaplnit zprávami, které jsou momentálně nepoužitelné, zatímco jiné úlohy mohou čekat na získání těchto zpráv. To má důsledky, pokud jde o úložiště, a pokud jde o podprocesy svázané s úlohami, které se pokoušejí získat zprávy.

Zpravidla, pokud máte více aplikací, které zpracovávají frontu, obvykle dosáhnete nejlepšího výkonu, když máte

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

pro každý synchronizační bod. Pokud frontu zpracovává pouze jedna aplikace, musíte mít pro každou pracovní jednotku více zpráv.

Pomocí atributu správce front **MAXUMSGS** můžete omezit počet zpráv, které může úloha získat nebo vložit v rámci jediné jednotky zotavení. Informace o tomto atributu viz příkaz **ALTER QMGR** v části Příkazy MQSC.

Výhody volání MQPUT1

Volání MQPUT1 použijte pouze v případě, že máte jednu zprávu pro vložení do fronty. Chcete-li vložit více než jednu zprávu, použijte volání MQOPEN následované řadou volání MQPUT a jedním voláním MQCLOSE .

Počet zpráv, které může správce front obsahovat

Lokální fronty

Počet lokálních zpráv, které může správce front zadržet, je v podstatě velikost sad stránek. Můžete mít až 100 sad stránek (ačkoli se doporučuje sada stránek 0 a sada stránek 1 jsou pro objekty a fronty související se systémem). Můžete použít sadu stránek s rozšířeným formátem a zvýšit kapacitu sady stránek.

Sdílené fronty

Kapacita sdílených front závisí na velikosti prostředku Coupling Facility (CF). Produkt IBM MQ používá struktury seznamu prostředku CF, kde základní jednotky úložiště jsou položky a prvky. Každá zpráva je uložena jako 1 položka a více prvků obsahujících přidružená data MQMD a další data zprávy. Počet prvků spotřebovaných jednou zprávou závisí na velikosti zprávy a pro CFLEVEL (5) na platných pravidlech odlehčování v době MQPUT. Při odlehčování dat zprávy do systému Db2 nebo SMDS je potřeba méně prvků. Přístup k datům zprávy je při odlehčování zprávy pomalejší. Další porovnání výkonu a režie CPU přidružené k odlehčování zpráv viz MP1H Performance Supportpac.

Co ovlivňuje výkon

Výkon může znamenat, jak rychle mohou být zprávy zpracovány, a může také znamenat, kolik CPU je potřeba na jednu zprávu.

Co ovlivňuje, jak rychle mohou být zprávy zpracovány

U trvalých zpráv je největším dopadem rychlost datových sad protokolu. Rychlost datových sad protokolu závisí na zařízení DASD, na kterém jsou. Proto je třeba dbát na to, aby se datová sada protokolu umístila na málo používaných svazcích, aby se snížilo soupeření. Rozložení protokolů produktu MQ zlepšuje výkon protokolu v případě, že je na jeden vstup/výstup zapsáno více stránek. Z High Performance Fibre Connection (zHPF) má také významný výkon pro dobu odezvy I/O, když je subsystém I/O zaneprázdněn.

Když existuje požadavek na získání a vložení zprávy, přístup do fronty je během požadavku uzamčen, aby byla zachována integrita fronty. Pro účely plánování zvažte uzamknutí fronty pro celý požadavek. Takže pokud je čas na vložení 100 mikrosekund a máte více než 10 000 požadavků za sekundu, může dojít k prodlevám. V praxi můžete dosáhnout lepších výsledků, ale je to dobré obecné pravidlo. Ke zlepšení výkonu můžete použít různé fronty.

Možné příčiny mohou být:

- použít společnou frontu odpovědí, kterou používá každá transakce CICS
- každá transakce CICS má jedinečnou odpověď do fronty
- odpověď na frontu pro oblast CICS a všechny transakce v oblasti CICS tuto frontu používají.

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

Pokud mají být zprávy čteny ze sady stránek, budou pomalejší v porovnání s tím, kdy jsou zprávy ve fondu vyrovnávacích pamětí. Pokud máte více zpráv, než se vejde do fondu vyrovnávacích pamětí, pak se vylijí na disk. Musíte tedy zajistit, aby fond vyrovnávacích pamětí byl dostatečně velký pro vaše zprávy s krátkou životností. Máte-li zprávy, které jste zpracovali o mnoho hodin později, je pravděpodobné, že se tyto zprávy vylijí na disk, takže byste měli očekávat, že tyto zprávy budou pomalejší, než kdyby byly ve fondu vyrovnávacích pamětí.

U 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 vytížení prostředku CF. Například na systémech Hursley, když byl CF 17% zaneprázdněn, byla doba odezvy 14 mikrosekund. Když byl prostředek CF zaneprázdněn 95%, byla doba odezvy 45 mikrosekund.

Pokud vaše požadavky produktu MQ používají velké množství CPU, může to ovlivnit rychlost zpracování zpráv. Protože pokud je logická oblast (LPAR) omezena na CPU, aplikace budou zpožděny při čekání na CPU.

Kolik CPU na zprávu

Obecně větší zprávy používají více CPU, takže pokud je to možné, vyhněte se velkým (x MB) zprávám.

Při získávání specifických zpráv z front by měla být fronta indexována, aby mohl správce front přejít přímo do zprávy (a tak se vyhnout potenciálně celému skenování fronty). Není-li fronta indexována, je procházena od začátku hledáním zprávy. Pokud je ve frontě 1000 zpráv, bude možná 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 pomocí řetězce selektoru kromě **CORRELID** nebo **MSGID**. Každá zpráva musí být prozkoumána, takže pokud je ve frontě mnoho zpráv, je to drahé.

Je efektivnější, když aplikace provádí OPEN PUT CLOSE než PUT1 PUT1.

Spouštění v CICS

Je-li rychlost příjmu zpráv pro spuštěnou frontu nízká, je efektivní použít nejprve spouštěč. Je-li rychlost příjmu zpráv více než 10 zpráv za sekundu, je efektivnější spustit první transakci, poté nechat transakci zpracovat zprávu a získat další zprávu atd. Pokud zpráva nedorazila v krátké době (například mezi 0.1 a 1 sekundou), transakce skončí. Při vysoké propustnosti můžete potřebovat více transakcí spuštěných ke zpracování zpráv a k zabránění vytváření zpráv. Pro každou vytvořenou zprávu spouštěče to vyžaduje vložení a získání zprávy spouštěče, což v podstatě zdvojnásobuje náklady na zprávu.

Počet podporovaných připojení nebo souběžných uživatelů

Každé připojení používá virtuální úložiště v rámci správce front, takže čím více souběžných uživatelů, tím více úložiště využívá. Pokud potřebujete velmi velký fond vyrovnávacích pamětí a velký počet uživatelů, můžete být omezeni pro virtuální úložiště a možná budete muset zmenšit velikost fondů vyrovnávacích pamětí.

Pokud je používáno zabezpečení, správce front po dlouhou dobu ukládá do mezipaměti informace v rámci správce front. Množství virtuálního úložiště, které je využito v rámci správce front, je ovlivněno.

Produkt **CHINIT** může podporovat až 10 000 připojení. Toto je omezeno virtuálním úložištěm. Pokud připojení používá více úložiště, například pomocí TLS, úložiště na připojení se zvýší, což znamená, že produkt **CHINIT** může podporovat méně připojení. Pokud zpracováváte velké zprávy, budou tyto zprávy vyžadovat více paměti pro vyrovnávací paměti v produktu **CHINIT**, aby produkt **CHINIT** mohl 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). V případě kanálu pro vzdáleného správce front může být před návratem odezvy přes síť odesláno 50 zpráv. Pokud uvažujete o rozsáhlé síti klientů, může být efektivnější používat správce front koncentrátoru v distribuovaném rámečku a mít jeden kanál přicházející do koncentrátoru a odvádějící z něj.

Další věci ovlivňující výkon

Datová sada protokolu by měla mít velikost alespoň 1000 cylindrů. Jsou-li protokoly menší než tento, může být aktivita kontrolního bodu příliš častá. Na vytíženém systému by měl být kontrolní bod obvykle každých 15 minut nebo déle, při velmi vysokých propustcích může být méně než tento. Při výskytu kontrolního bodu jsou fondy vyrovnávacích pamětí skenovány a 'staré' zprávy a změněné stránky jsou zapsány na disk. Pokud jsou kontrolní body příliš časté, může to ovlivnit výkon. Hodnota parametru LOGLOAD může také ovlivnit frekvenci kontrolních bodů. Pokud dojde k nestandardnímu ukončení správce front, je možné, že při restartu bude nutné provést načtení zpět do 3 kontrolních bodů. Nejlepším intervalem kontrolního bodu je rovnováha mezi aktivitou při vytvoření kontrolního bodu a množstvím dat protokolu, která mohou být při restartování správce front čtena.

Při spouštění kanálu existuje značná režie. Obvykle je lepší spustit kanál a nechat jej připojený, spíše než časté spouštění a zastavení kanálu.

Související informace

MP1K: IBM MQ for z/OS 9.0 Sestava výkonu

z/OS IMS a IMS přemostění aplikací na IBM MQ for z/OS

Tyto informace vám pomohou psát aplikace IMS pomocí IBM MQ.

- Chcete-li používat synchronizační body a volání MQI v aplikacích IMS , viz “Psaní aplikací IMS pomocí IBM MQ” na stránce 66.
- Chcete-li psát aplikace, které používají most IBM MQ - IMS , viz “Zápis aplikací mostu IMS” na stránce 70.

Další informace o aplikacích mostu IMS a IMS v systému IBM MQ for z/OSZískáte pomocí následujících odkazů:

- “Psaní aplikací IMS pomocí IBM MQ” na stránce 66
- “Zápis aplikací mostu IMS” na stránce 70

Související pojmy

“Přehled rozhraní fronty zpráv” na stránce 700
Informace o komponentách rozhraní MQI (Message Queue Interface).

“Připojení ke správci front a odpojení od něj” na stránce 713
Chcete-li používat programovací služby IBM MQ , musí mít program připojení ke správci front. Pomocí těchto informací se dozvíte, jak se připojit ke správci front a jak se od něj odpojit.

“Otevírání a zavírání objektů” na stránce 720
Tyto informace poskytují náhled na otevírání a zavírání objektů IBM MQ .

“Vkládání zpráv do fronty” na stránce 730
Pomocí těchto informací se dozvíte, jak vkládat zprávy do fronty.

“Získávání zpráv z fronty” na stránce 744
Pomocí těchto informací získáte informace o získávání zpráv z fronty.

“Zjišťování a nastavení atributů objektu” na stránce 822
Atributy jsou vlastnosti, které definují charakteristiky objektu IBM MQ .

“Potvrzení a zálohování jednotek práce” na stránce 825
Tyto informace popisují, jak potvrdit a vrátit zpět všechny obnovitelné operace get a put, které se vyskytly v transakci.

“Spuštění aplikací IBM MQ pomocí spouštěčů” na stránce 836
Získejte informace o spouštěčích a o tom, jak spouštět aplikace IBM MQ pomocí spouštěčů.

“Práce s rozhraním MQI a klastry” na stránce 854
Existují speciální volby pro volání a návratové kódy, které se týkají klastrování.

“Používání a psaní aplikací na systému IBM MQ for z/OS” na stránce 858
Aplikace IBM MQ for z/OS lze vytvořit z programů, které jsou spuštěny v mnoha různých prostředích. To znamená, že mohou využívat zařízení, která jsou k dispozici ve více než jednom prostředí.

z/OS Psaní aplikací IMS pomocí IBM MQ

Při používání produktu IBM MQ v aplikacích IMS je třeba vzít v úvahu další aspekty. Patří mezi ně volání rozhraní API produktu MQ a mechanizmus používaný pro synchronizační bod.

Chcete-li se dozvědět více o psaní aplikací IMS na webu IBM MQ for z/OS, použijte následující odkazy:

- “Synchronizační body v aplikacích IMS” na stránce 67
- “Volání MQI v aplikacích IMS” na stránce 67

Omezení

Existují omezení, která mohou volání rozhraní API produktu IBM MQ používat aplikace používající adaptér IMS .

Následující volání rozhraní API IBM MQ nejsou podporována v rámci aplikace používající adaptér IMS :

- MQCB
- MQCB_FUNCTION
- MQCTL

Související pojmy

“Zápis aplikací mostu IMS” na stránce 70

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

z/OS *Synchronizační body v aplikacích IMS*

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

Chcete-li vrátit zpět všechny změny od předchozího kontrolního bodu, můžete použít volání IMS ROLB (odvolání). Další informace viz Volání ROLB v dokumentaci k produktu IMS .

Správce front je účastníkem protokolu dvoufázového potvrzování; koordinátorem je správce synchronizačních bodů IMS .

Všechny otevřené popisovače jsou uzavřeny adaptérem IMS v synchronizačním bodu (s výjimkou prostředí BMP s dávkovým zpracováním nebo bez použití zpráv). Důvodem je skutečnost, že při provádění volání MQCONN, MQCONNX a MQOPEN může jiný uživatel zahájit další transakci a při provádění volání MQPUT nebo MQGET je provedena kontrola zabezpečení produktu IBM MQ .

Avšak v prostředí WFI (Wait-for-Input) nebo PWFI (pseudo-Wait-for-Input) IMS neoznámí IBM MQ zavření popisovačů, dokud nedorazí další zpráva nebo dokud se aplikaci nevrátí stavový kód QC. Pokud aplikace čeká v oblasti IMS a některý z těchto manipulátorů patří do spouštěných front, nedojde ke spuštění, protože jsou fronty otevřené. Z tohoto důvodu by aplikace spuštěné v prostředí WFI nebo PWFI měly explicitně MQCLOSE manipulovat s frontou před provedením GU pro IOPCB pro další zprávu.

Pokud aplikace IMS (buď BMP, nebo MPP) vydá volání MQDISC, otevřené fronty se zavřou, ale neprovede se žádný implicitní synchronizační bod. Pokud aplikace skončí normálně, všechny otevřené fronty se zavřou a dojde k implicitnímu potvrzení. Pokud aplikace skončí abnormálně, všechny otevřené fronty se zavřou a dojde k implicitnímu vrácení.

z/OS *Volání MQI v aplikacích IMS*

Pomocí těchto informací získáte informace o používání volání MQI v aplikacích Server a v aplikacích Poptávka.

Tento oddíl se zabývá použitím volání MQI v následujících typech aplikací IMS :

- “Serverové aplikace” na stránce 67
- “Žádosti o dotazování” na stránce 70

Serverové aplikace

Zde je uveden přehled 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 produktu BMP používajícího tento model v produktu C/370. Program nejprve naváže komunikaci s produktem IMS a poté s produktem 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 produktu IMS určuje, zda byl program volán jako objekt BMP řízený zprávami nebo dávkově orientovaný, a řídí odpovídajícím způsobem připojení a obslužné rutiny front produktu IBM MQ :

```
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 produktu IBM MQ se připojí ke správci front a otevře fronty. V BMP řízeném zprávami je toto voláno po každém synchronizačním bodu IMS ; v dávkově orientovaném BMP je toto voláno pouze při spuštění 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 jednu pracovní jednotku pro každé vyvolání. Důvodem je skutečnost, že při použití synchronizačního bodu (GU) jsou uzavřeny manipulátory připojení a fronty a je doručena další zpráva produktu IMS . Toto omezení lze částečně překonat jedním z následujících způsobů:

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

  Jedná se o:

  – Čtení zprávy
  – Zpracování požadovaných aktualizací
  – Vložení odpovědi

  ve smyčce, dokud nebudou zpracovány všechny zprávy nebo dokud nebude zpracován nastavený maximální počet zpráv, při kterém se provede synchronizační bod.

  Tímto způsobem lze přistupovat pouze k určitým typům aplikací (například k jednoduché aktualizaci databáze nebo dotazu). Ačkoli lze zprávy odpovědí rozhraní MQI vložit s oprávněním původce zpracovávané zprávy rozhraní MQI, je třeba pečlivě řešit dopady na zabezpečení všech aktualizací prostředků produktu IMS .

- **Zpracování jedné zprávy na vyvolání MPP a zajištění vícenásobného plánování MPP pro zpracování všech dostupných zpráv.**

  Pomocí programu pro monitorování spouštěčů IBM MQ IMS (CSQQTRMN) naplánujte transakci MPP, když jsou ve frontě IBM MQ zprávy a žádné aplikace ji neobsluhují.

  Pokud monitor spouštěčů spustí protokol MPP, předá název správce front a název fronty programu, jak ukazuje následující extrakce kódu v jazyce 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, u kterého se očekává, že se jedná o dlouhotrvající úlohu, je lépe podporován v oblasti dávkového zpracování, ačkoli BMP nelze spustit pomocí CSQQTRMN.

Žádosti o dotazování

Typická aplikace systému IBM MQ , která zahajuje dotaz nebo aktualizaci, funguje takto:

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

Vzhledem k tomu, že zprávy vkládané do front systému IBM MQ nejsou zpřístupněny jiným aplikacím systému IBM MQ , dokud nejsou potvrzeny, musí být buď umístěny mimo synchronizační bod, nebo musí být aplikace IMS rozdělena do dvou transakcí.

Pokud dotaz zahrnuje vložení jediné zprávy, můžete použít volbu *no syncpoint* ; pokud je však dotaz složitější nebo pokud jsou zahrnuty aktualizace prostředků, může dojít k problémům s konzistencí, pokud dojde k selhání a nepoužijete funkci syncpoint.

Chcete-li to překonat, můžete rozdělit transakce produktu IMS MPP pomocí volání MQI pomocí přepínače zpráv program-program; informace o tom viz *IMS ISC (Intersystem Communication)* . To umožňuje implementaci dotazovacího programu 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
```

z/OS 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 IMS.

Pomocí následujících odkazů získáte další informace o psaní aplikací mostu IMS v systému IBM MQ for z/OS:

- “Jak most IMS pracuje se zprávami” na stránce 71
- “Zápis IMS transakčních programů prostřednictvím IBM MQ” na stránce 880

Související pojmy

“Psaní aplikací IMS pomocí IBM MQ” na stránce 66
Při používání produktu IBM MQ v aplikacích IMS je třeba vzít v úvahu další aspekty. Patří mezi ně volání rozhraní API produktu MQ a mechanizmus používaný pro synchronizační bod.

z/OS *Jak most IMS pracuje se zprávami*

Používáte-li most IBM MQ - IMS k odesílání zpráv do aplikace IMS , je třeba sestavit zprávy ve speciálním formátu.

Zprávy musíte také vložit do front systému IBM MQ , které byly definovány s paměťovou třídou, která určuje skupinu XCF a název člena cílového systému IMS . Tyto fronty se nazývají MQ-IMS nebo jednoduše **fronty mostu** .

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

Uživatel se nemusí přihlásit k produktu IMS před odesláním zpráv do aplikace IMS . ID uživatele v poli *UserIdentifier* struktury MQMD se používá pro kontrolu zabezpečení. Úroveň kontroly je určena při připojení produktu IBM MQ k serveru IMSa je popsána v tématu Řízení přístupu k aplikaci pro most IMS. To umožňuje, aby bylo implementováno pseudopřihlášení.

Most IBM MQ - IMS přijímá následující typy zpráv:

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

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

  Poznámka:

  1. Hranaté závorky [] představují volitelné vícenásobné segmenty.
  2. Nastavte pole *Format* struktury MQMD na MQFMT_IMS pro použití struktury MQIIH.

- Zprávy obsahující data transakce IMS , ale bez struktury MQIIH:

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

Produkt IBM MQ ověří data zprávy a ujistí se, že součet bajtů LL plus délka MQIIH (je-li přítomna) se rovná délce zprávy.

Když most IBM MQ - IMS získá zprávy z front mostu, zpracuje je takto:

- Pokud zpráva obsahuje strukturu MQIIH, most ověří MQIIH (viz MQIIH), sestaví záhlaví OTMA a odešle zprávu do IMS. Kód transakce je uveden ve vstupní zprávě. Pokud se jedná o LTERM, produkt IMS odpoví zprávou DFS1288E . Pokud kód transakce představuje příkaz, IMS provede příkaz; jinak se zpráva zařadí do fronty IMS pro transakci.
- Pokud zpráva obsahuje transakční data IMS , ale nemá strukturu MQIIH, most IMS vytvoří následující předpoklady:
  - Kód transakce je v bajtech 5 až 12 uživatelských dat
  - Transakce je v nekonverzačním režimu
  - Transakce je v režimu potvrzení 0 (commit-then-send)
  - Parametr *Format* v deskriptoru MQMD se používá jako parametr *MFSMapName* (na vstupu).
  - Režim zabezpečení je MQISS_CHECK

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

Most IBM MQ - IMS používá pro každou frontu IBM MQ jednu nebo dvě propojení procesů:

- Synchronizovaná propojení procesů se používá pro všechny zprávy používající režim potvrzení 0 (COMMIT_THEN_SEND) (tyto zprávy se zobrazují s hodnotou SYN v poli stavu příkazu IMS /DIS TMEMBER client TPIPE xxxx).
- Nesynchronizovaná Tpipe se používá pro všechny zprávy používající režim potvrzení 1 (SEND_THEN_COMMIT)

Propojení procesů jsou vytvořena produktem IBM MQ při prvním použití. Nesynchronizované propojení procesů existuje, dokud není produkt IMS restartován. Synchronizovaná propojení procesů existují až do studeného spuštění produktu IMS . Tyto propojení procesů nemůžete odstranit sami.

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

- “Mapování zpráv IBM MQ na typy transakcí IMS” na stránce 72
- “Pokud zprávu nelze vložit do fronty IMS” na stránce 72
- “Kódy zpětné vazby mostu IMS” na stránce 73
- “Pole MQMD ve zprávách z mostu IMS” na stránce 73
- “Pole MQIIH ve zprávách z mostu IMS” na stránce 74
- “Odpovědět na zprávy od IMS” na stránce 75
- “Použití alternativních PCB odpovědi v transakcích IMS” na stránce 75
- “Odesílání nevyžádaných zpráv z IMS” na stránce 76
- “Segmentace zpráv” na stránce 76
- “Převod dat pro zprávy do a z mostu IMS” na stránce 76

Související pojmy

“Zápis IMS transakčních programů prostřednictvím IBM MQ” na stránce 880

Kódování požadované pro zpracování IMS transakcí prostřednictvím IBM MQ závisí na formátu zprávy požadovaném transakcí IMS a rozsahu odpovědí, které může vrátit. Existuje však několik bodů, které je třeba zvážit, když vaše aplikace zpracovává informace o formátování obrazovky IMS .

z/OS *Mapování zpráv IBM MQ na typy transakcí IMS*

Tabulka popisující mapování zpráv IBM MQ na typy transakcí IMS .

Tabulka 4. Způsob mapování zpráv produktu IBM MQ na typy transakcí IMS

IBM MQ typ zprávy	**Commit-then-send (režim 0)- používá synchronizované IMS Tpipes**	**Send-then-commit (režim 1)- používá nesynchronizované IMS Tpipes**
Trvalé zprávy IBM MQ	• Zotavitelné transakce s plnými funkcemi • Nezotavitelné transakce jsou odmítnuty IMS	• Rychlé transakce • Konverzační transakce • Transakce s plnou funkcí
Přechodné zprávy IBM MQ	• Nezotavitelné transakce s plnými funkcemi • Obnovitelné transakce jsou povoleny s produktem IMS V8 a opravou APAR PQ61404 a všemi novějšími verzemi produktu IMS	• Rychlé transakce • Konverzační transakce • Transakce s plnou funkcí

Poznámka: Příkazy IMS nemohou používat trvalé zprávy IBM MQ s režimem potvrzení 0. Další informace viz Režim potvrzení (commitMode) .

z/OS *Pokud zprávu nelze vložit do fronty IMS*

Informace o akcích, které je třeba provést v případě, že zprávu nelze vložit do fronty IMS .

Pokud zprávu nelze vložit do fronty IMS , IBM MQprovede následující akci:

- Pokud zprávu nelze vložit do souboru IMS , protože je neplatná, je vložena do fronty nedoručených zpráv a zpráva je odeslána do systémové konzoly.

- Pokud je zpráva platná, ale je odmítnuta produktem IMS, produkt IBM MQ odešle chybovou zprávu na systémovou konzolu, zpráva obsahuje chybový kód IMS a zpráva IBM MQ je vložena do fronty nedoručených zpráv. Je-li IMS chybový kód 001A, IMS odešle zprávu IBM MQ obsahující příčinu selhání do fronty pro odpověď.

  Poznámka: Za výše uvedených okolností platí, že pokud produkt IBM MQ nemůže z jakéhokoli důvodu vložit zprávu do fronty nedoručených zpráv, vrátí se zpráva do původní fronty IBM MQ . Do systémové konzoly se odešle chybová zpráva a z této fronty se neodešlou žádné další zprávy.

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

  - Zastavte a restartujte propojení procesů v adresáři IMS odpovídající frontě.
  - Změnit frontu na GET (DISABLED) a znovu na GET (ENABLED)
  - Zastavte a restartujte IMS nebo OTMA
  - Zastavte a restartujte subsystém IBM MQ .
- Pokud je zpráva odmítnuta produktem IMS kvůli jiné chybě než kvůli chybě zprávy, vrátí se zpráva IBM MQ do původní fronty, produkt IBM MQ zastaví zpracování fronty a na systémovou konzolu se odešle chybová zpráva.

  Je-li vyžadována zpráva sestavy výjimek, most ji vloží do fronty pro odpověď s oprávněním původce. Pokud zprávu nelze vložit do fronty, zpráva sestavy se vloží do fronty nedoručených zpráv s oprávněním mostu. Pokud jej nelze vložit do fronty DLQ, bude vyřazen.

z/OS *Kódy zpětné vazby mostu IMS*

IMS chybové kódy jsou obvykle vypisovány v hexadecimálním formátu ve zprávách konzoly IBM MQ , jako např. CSQ2001I (například chybový kód 0x001F). Kódy zpětné vazby IBM MQ uvedené v záhlaví nedoručených zpráv vložených do fronty nedoručených zpráv jsou desetinná čísla.

Kódy zpětné vazby mostu IMS jsou v rozsahu 301 až 399 nebo 600 až 855 pro chybový kód NACK 0x001A. Jsou mapovány ze smyslových kódů IMS-OTMA takto:

1. Kód chyby IMS-OTMA je převeden z hexadecimálního čísla na desetinné číslo.
2. 300 se přidá k číslu, které je výsledkem výpočtu v 1, s uvedením kódu IBM MQ *Feedback* .
3. Chybový kód IMS-OTMA 0x001A, desetinný 26 je speciální případ. Vygeneruje se kód *Zpětná vazba* v rozsahu 600-855.
   a. Kód příčiny IMS-OTMA je převeden z hexadecimálního čísla na desetinné číslo.
   b. 600 se přidá k číslu, které je výsledkem výpočtu v a, s kódem IBM MQ *Zpětná vazba* .

Informace o chybových kódech IMS-OTMA viz chybové kódy OTMA pro zprávy NAK.

z/OS *Pole MQMD ve zprávách z mostu IMS*

Informace o polích MQMD ve zprávách z mostu IMS .

Modul MQMD původní zprávy je přenášen produktem IMS v sekci Uživatelská data záhlaví OTMA. Pokud zpráva pochází z produktu IMS, je sestavena uživatelskou procedurou pro rozlišení cíle systému IMS . Modul MQMD zprávy přijaté z produktu IMS je sestaven následujícím způsobem:

StrucID
"MD"

Verze
MQMD_VERSION_1

Sestava
MQRO_NONE

MsgType
MQMT_REPLY

Vypršení
Pokud je v poli Příznaky MQIIH nastavena hodnota MQIH_PASS_EXPIRATION, bude toto pole obsahovat zbývající čas vypršení platnosti, jinak bude nastaveno 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, pokud je MQMD.Format vstupní zprávy je MQFMT_IMS, jinak IOPCB.MODNAME

Priorita
MQMD.Priority vstupní zprávy

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

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

CorrelId
MQMD.CorrelId ze vstupní zprávy, pokud MQRO_PASS_CORREL_ID, jinak MQMD.MsgId ze vstupní zprávy (výchozí)

BackoutCount
0

ReplyToQ
Mezery

ReplyToQMgr
Mezery (správce front během operace MQPUT nastavil na lokální název správce front)

UserIdentifier
MQMD.UserIdentifier vstupní zprávy

AccountingToken
MQMD.AccountingToken vstupní zprávy

ApplIdentityData
MQMD.ApplIdentityData vstupní zprávy

PutApplType
MQAT_XCF-pokud nedošlo k chybě, jinak MQAT_BRIDGE

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

PutDate
Datum, kdy byla zpráva vložena

PutTime
Čas, kdy byla zpráva vložena

ApplOriginData
Mezery

z/OS *Pole MQIIH ve zprávách z mostu IMS*

Získejte informace o polích MQIIH ve zprávách z mostu IMS .

MQIIH zprávy přijaté od IMS je sestaveno takto:

StrucId
"IIH"

Verze
1

StrucLength
84

Kódování
MQENC_NATIVE

CodedCharSetId
MQCCSI_Q_MGR

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

Příznaky
0

LTermOverride
Název LTERM (Tpipe) ze záhlaví OTMA

MFSMapName
Název mapy ze záhlaví OTMA

ReplyToFormát
Mezery

Ověřovatel
MQIIH.Authenticator vstupní zprávy, pokud je zpráva odpovědi vkládána do fronty mostu MQ-IMS , jinak prázdná hodnota.

ID TranInstance
ID konverzace/Token serveru ze záhlaví OTMA, pokud je v konverzaci. Ve verzích systému IMS starších než V14je toto pole vždy prázdné, pokud není v konverzaci. Od verze IMS V14 dále může toto pole nastavit IMS , i když není v konverzaci.

TranState
"C", pokud je v konverzaci, jinak prázdné

CommitMode
Režim potvrzení ze záhlaví OTMA ("0" nebo "1")

SecurityScope
Prázdný

Vyhrazeno
Prázdný

z/OS *Odpovědět na zprávy od IMS*

Když transakce IMS ISRTs na svůj IOPCB, zpráva je směrována zpět na původní LTERM nebo TPIPE.

Ty jsou v produktu IBM MQ vnímány jako zprávy odpovědi. Zprávy odpovědí z produktu IMS jsou vloženy do fronty pro odpověď určené v původní zprávě. Pokud zprávu nelze vložit do fronty pro odpověď, je vložena do fronty nedoručených zpráv s použitím oprávnění mostu. Pokud zprávu nelze vložit do fronty nedoručených zpráv, odešle se na adresu IMS negativní potvrzení, že zprávu nelze přijmout. Odpovědnost za zprávu se pak vrátí na IMS. Používáte-li režim potvrzení 0, nebudou zprávy z tohoto propojení procesů odesílány do mostu a zůstanou ve frontě IMS . To znamená, že do restartování nebudou odesílány žádné další zprávy. Pokud používáte režim potvrzení 1, může pokračovat jiná práce.

Pokud má odpověď strukturu MQIIH, její typ formátu je MQFMT_IMS; pokud ne, její typ formátu je určen názvem IMS MOD použitým při vkládání zprávy.

z/OS *Použití alternativních PCB odpovědi v transakcích IMS*

Když transakce IMS používá alternativní PCB odezvy (ISRTs pro ALTPCB, nebo vydá volání CHNG pro modifikovatelný PCB), je vyvolána uživatelská procedura před směrováním (DFSYPRX0), aby se zjistilo, zda má být zpráva přesměrována.

Pokud má být zpráva přesměrována, je vyvolána uživatelská procedura rozpoznání místa určení (DFSYDRU0), která potvrdí místo určení a připraví informace v záhlaví. Informace o těchto uživatelských programech naleznete v tématu Použití uživatelských procedur OTMA v produktu IMS a Uživatelský program před směrováním DFSYPRX0 .

Není-li v uživatelských procedur provedena akce, všechny výstupy z transakcí IMS zahájených ze správce front IBM MQ , ať už do IOPCB nebo ALTPCB, budou vráceny do stejného správce front.

z/OS *Odesílání nevyžádaných zpráv z IMS*

Chcete-li odeslat zprávy z IMS do fronty IBM MQ , musíte vyvolat transakci IMS , kterou ISRTs odešle do ALTPCB.

Pro směrování nevyžádaných zpráv z produktu IMS a sestavení uživatelských dat OTMA je třeba napsat uživatelské procedury pro předběžné směrování a rozlišení cíle, aby bylo možné správně sestavit MQMD zprávy. Informace o těchto uživatelských programech viz Uživatelská procedura před směrováním DFSYPRX0 a Uživatelská procedura pro rozlišení cíle .

Poznámka: Most IBM MQ - IMS neví, zda je zpráva, kterou obdrží, odpovědí nebo nevyžádanou zprávou. Zpracovává zprávu stejným způsobem v každém případě, sestaví MQMD a MQIIH odpovědi na základě UserData OTMA, která přišla se zprávou.

Nevyžádané zprávy mohou vytvářet nové Troury. Pokud například existující transakce IMS přepnula na nový LTERM (například PRINT01), ale implementace vyžaduje, aby byl výstup doručen prostřednictvím OTMA, vytvoří se nový Tpipe (v tomto příkladu s názvem PRINT01). Standardně se jedná o nesynchronizované propojení procesů. Pokud implementace vyžaduje, aby byla zpráva obnovitelná, nastavte příznak výstupu ukončení rozpoznání cíle. Další informace viz příručka *IMS Customization Guide* .

z/OS *Segmentace zpráv*

Transakce IMS můžete definovat jako očekávaný vstup s jedním nebo více segmenty.

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

Maximální délka datového segmentu LLZZ je definována pomocí IMS/OTMA (32767 bajtů). Celková délka zprávy IBM MQ je součtem bajtů LL plus délka struktury MQIIH.

Všechny segmenty odpovědi jsou obsaženy v jediné zprávě IBM MQ .

Existuje další omezení omezení 32 kB pro zprávy s formátem MQFMT_IMS_VAR_STRING. Když jsou data ve zprávě ASCII-mixed CCSID převedena na zprávu EBCDIC-mixed CCSID, je shift-in byte nebo shift-out byte přidán pokaždé, když dojde k přechodu mezi znaky SBCS a DBCS. Omezení 32 kB platí pro maximální velikost zprávy. To znamená, že protože pole LL ve zprávě nesmí překročit 32 kB, nesmí zpráva překročit 32 kB včetně všech znaků shift-in a shift-out. Aplikace, která zprávu sestavuje, to musí povolit.

z/OS *Převod dat pro zprávy do a z mostu IMS*

Převod dat provádí buď distribuovaný prostředek front (který může volat nezbytné uživatelské procedury), nebo agent front v rámci skupiny (který nepodporuje použití uživatelských procedur), když vkládá zprávu do cílové fronty, která má pro svou paměťovou třídu definovány informace XCF. K převodu dat nedojde, když je zpráva doručena do fronty prostřednictvím publikování/odběru.

Všechny potřebné uživatelské procedury musí být k dispozici pro službu distribuovaného řazení do 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 .

Dojde-li k chybám při převodu, zpráva se vloží do fronty nepřevedené; to má za následek, že most IBM MQ - IMS bude nakonec považován za chybu, protože most nerozpozná formát záhlaví. Dojde-li k chybě převodu, odešle se chybová zpráva na konzolu z/OS .

Podrobné informace o převodu dat obecně viz “Zápis uživatelských procedur převodu dat” na stránce 950 .

Odesílání zpráv do mostu IBM MQ - IMS

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

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

Pokud jsou vaše data (jiná než LLZZs) všechna znaková data (MQCHAR), použijte jako název formátu (v MQIIH nebo MQMD, podle potřeby) vestavěný formát MQFMT_IMS_VAR_STRING. Jinak použijte vlastní název formátu, v takovém případě musíte také poskytnout uživatelskou proceduru pro převod dat pro váš formát. Uživatelská procedura musí zpracovat převod LLZZs ve vaší zprávě, kromě dat samotných (ale nemusí zpracovat žádnou MQIIH na začátku zprávy).

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

Příjem zpráv z mostu IBM MQ - IMS

Je-li v původní zprávě, kterou odesíláte do produktu IMS, uvedena struktura MQIIH, je uvedena i ve zprávě odpovědi.

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

- Máte-li strukturu MQIIH pro původní zprávu, zadejte požadovaný formát pro zprávu odpovědi do pole MQIIH *ReplytoFormat* původní zprávy. Tato hodnota je umístěna do pole MQIIH *Format* zprávy odpovědi. To je užitečné zejména v případě, že všechna výstupní data jsou ve formátu LLZZ < znaková data >.
- Pokud nemáte strukturu MQIIH ve své původní zprávě, uveďte formát, který chcete pro zprávu odpovědi, jako název MFS MOD v ISRT aplikace IMS pro IOPCB.

Vývoj aplikací JMS/Jakarta Messaging a Java

Produkt IBM MQ poskytuje tři Java jazyková rozhraní: IBM MQ classes for Jakarta Messaging, IBM MQ classes for JMSa IBM MQ classes for Java.

Informace o této úloze

V 9.3.0 JM 3.0 V 9.3.0 **IBM MQ classes for Jakarta Messaging**

IBM MQ classes for Jakarta Messaging je poskytovatel systému Jakarta Messaging , který implementuje rozhraní Jakarta Messaging pro systém zpráv IBM MQ . Produkt Jakarta Connectors Architecture poskytuje standardní způsob připojení aplikací spuštěných v prostředí Jakarta EE k podnikovému informačnímu systému (EIS), například IBM MQ nebo Db2.

Další informace naleznete v tématech “Proč bych měl používat IBM MQ classes for Jakarta Messaging?” na stránce 79 a “Přístup k IBM MQ z Java -Volba rozhraní API” na stránce 82.

JMS 2.0 **IBM MQ classes for JMS**

IBM MQ classes for JMS je poskytovatel systému JMS , který implementuje rozhraní JMS pro systém zpráv IBM MQ . Java Platform, Enterprise Edition Connector Architecture (JCA) poskytuje standardní způsob připojení aplikací spuštěných v prostředí Java EE k podnikovému informačnímu systému (EIS), například IBM MQ nebo Db2.

Další informace naleznete v tématech “Proč bych měl používat IBM MQ classes for JMS?” na stránce 80 a “Přístup k IBM MQ z Java -Volba rozhraní API” na stránce 82.

IBM MQ classes for Java

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

Produkt IBM MQ classes for Java zapouzdřuje rozhraní MQI (Message Queue Interface), nativní rozhraní API systému IBM MQ , a používá stejný model objektů jako jiná objektově orientovaná rozhraní, zatímco IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging implementují Java rozhraní systému zpráv z produktu Oracle a Java Community Process .

Další informace naleznete v tématech “Proč bych měl používat IBM MQ classes for Java?” na stránce 336 a “Přístup k IBM MQ z Java -Volba rozhraní API” na stránce 82.

Poznámka:

Stabilized Produkt IBM neprovede žádná další vylepšení produktu IBM MQ classes for Java a jsou funkčně stabilizovány na úrovni dodané v produktu IBM MQ 8.0. Existující aplikace, které používají produkt IBM MQ classes for Java , jsou i nadále plně podporovány, ale nové funkce nebudou přidány a požadavky na vylepšení budou odmítnuty. Plně podporováno znamená, že defekty budou opraveny společně se změnami, které jsou vynucené změnami systémových požadavků IBM MQ .

IBM MQ classes for Java nejsou v produktu IMSpodporovány.

IBM MQ classes for Java nejsou v produktu WebSphere Libertypodporovány. Nesmí být použity s funkcí systému zpráv IBM MQ Liberty ani s generickou podporou systému JCA . Další informace naleznete v tématu Použití rozhraní WebSphere MQ Java v prostředí J2EE/JEE.

Použití IBM MQ classes for JMS/Jakarta Messaging

IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging jsou poskytovatelé systému zpráv Java dodávaní s produktem IBM MQ. Kromě implementace rozhraní definovaných ve specifikacích JMS a Jakarta Messaging tito poskytovatelé systému zpráv přidávají do rozhraní API systému zpráv Java dvě sady rozšíření.

V 9.3.0 JM 3.0 V 9.3.0 V produktu IBM MQ 9.3.0je produkt Jakarta Messaging 3.0 podporován pro vývoj nových aplikací. Produkt IBM MQ 9.3.0 nadále podporuje produkt JMS 2.0 pro existující aplikace. Použití rozhraní API JMS 2.0 a rozhraní API Jakarta Messaging 3.0 ve stejné aplikaci není podporováno.

Poznámka: V případě systému Jakarta Messaging 3.0se řízení specifikace JMS přesune z Oracle do Java Community Process. Produkt Oracle si však zachovává řízení názvu "javax", který se používá v jiných technologiích Java , jež nebyly přesunuty do procesu komunity Java . Takže zatímco Jakarta Messaging 3.0 je funkčně ekvivalentní produktu JMS 2.0 , existují určité rozdíly v pojmenování:

- Oficiální název pro verzi 3.0 je Jakarta Messaging spíše než Java Message Service.
- Názvy balíků a konstant mají předponu `jakarta` spíše než `javax`. Například v produktu JMS 2.0 je počátečním připojením k poskytovateli systému zpráv objekt `javax.jms.Connection` a v produktu Jakarta Messaging 3.0 objekt `jakarta.jms.Connection` .

JMS 2.0 Balíky javax.jms definují rozhraní 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.

JM 3.0 Balíky jakarta.jms definují rozhraní Jakarta Messaging a poskytovatel Jakarta Messaging implementuje tato rozhraní pro specifický produkt systému zpráv. IBM MQ classes for Jakarta Messaging je poskytovatel Jakarta Messaging , který implementuje rozhraní Jakarta Messaging pro IBM MQ.

Specifikace JMS a Jakarta Messaging očekávají, že objekty ConnectionFactory a objekty Destination budou spravovány. Administrátor vytvoří a udržuje spravované objekty v centrálním úložišti a aplikace JMS nebo Jakarta Messaging tyto objekty načte pomocí Java Naming Directory Interface (JNDI).

JMS 2.0 V případě systému JMS 2.0může administrátor použít nástroj pro administraci IBM MQ JMS **`JMSAdmin`**nebo IBM MQ Explorerk vytvoření a údržbě spravovaných objektů v centrálním úložišti.

JM 3.0 Pro Jakarta Messaging 3.0nemůžete spravovat rozhraní JNDI pomocí IBM MQ Explorer. Administrace rozhraní JNDI je podporována Jakarta Messaging 3.0 variantou **JMSAdmin**, což je **JMS30Admin**.

Protože JMS a Jakarta Messaging sdílejí mnoho společného, lze další odkazy na JMS v tomto tématu považovat za odkazy na oboje. Případné rozdíly jsou podle potřeby zvýrazněny.

Produkt IBM MQ classes for JMS také poskytuje dvě sady rozšíření rozhraní API JMS . Hlavní zaměření těchto rozšíření se týká vytváření a konfigurace továren připojení a míst určení dynamicky za běhu, ale tato rozšíření také poskytují funkci, která přímo nesouvisí se systémem zpráv, například funkci pro určování problémů.

Rozšíření produktu IBM MQ JMS
IBM MQ classes for JMS obsahuje rozšíření, která jsou implementována v objektech, jako jsou objekty MQConnectionFactory, MQQueue a MQTopic. Tyto objekty mají vlastnosti a metody specifické pro IBM MQ. Objekty mohou být spravovány objekty, nebo aplikace může vytvářet objekty dynamicky za běhu. Tato rozšíření se nazývají rozšíření produktu IBM MQ JMS .

Rozšíření produktu IBM JMS
Produkt IBM MQ classes for JMS také poskytuje obecnější sadu rozšíření rozhraní JMS API, která nejsou specifická pro IBM MQ jako systém zpráv nebo Java jako používaný programovací jazyk. Tato rozšíření se nazývají rozšíření produktu IBM JMS a mají následující obecné cíle:

- Chcete-li zajistit vyšší úroveň konzistence mezi poskytovateli produktu IBM JMS .
- Usnadnit zápis aplikace mostu mezi dvěma systémy systému zpráv IBM .
- Usnadnit portování aplikace z jednoho poskytovatele produktu IBM JMS na jiného.

Rozšíření poskytují funkci, která je podobná funkci poskytnuté v IBM MQ Message Service Client (XMS) for C/C++ a IBM MQ Message Service Client (XMS) for .NET.

Související pojmy
Jazyková rozhraní produktu IBM MQ Java
Související úlohy
“Psaní aplikací IBM MQ classes for JMS/Jakarta Messaging” na stránce 135
Po stručném úvodu k modelu JMS tato sekce poskytuje podrobné pokyny, jak psát aplikace IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging .

V 9.3.0 JM 3.0 V 9.3.0 Proč bych měl používat IBM MQ classes for Jakarta Messaging?

Použití produktu IBM MQ classes for Jakarta Messaging má řadu výhod, včetně možnosti znovu použít existující odbornosti produktu Jakarta Messaging ve vaší organizaci, a aplikací, které jsou více nezávislé na poskytovateli Jakarta Messaging a základní konfiguraci produktu IBM MQ .

Přehled výhod použití IBM MQ classes for Jakarta Messaging

Použití produktu IBM MQ classes for Jakarta Messaging vám umožňuje znovu použít existující odbornosti Jakarta Messaging a zajistit nezávislost aplikace.

- Můžete znovu použít odbornosti Jakarta Messaging .

  IBM MQ classes for Jakarta Messaging je poskytovatel systému Jakarta Messaging , který implementuje rozhraní Jakarta Messaging pro systém zpráv IBM MQ . Pokud je vaše organizace v produktu IBM MQnová, ale již má znalosti vývoje Jakarta Messaging (nebo JMS) aplikací, může být snazší používat známé rozhraní Jakarta Messaging API pro přístup k prostředkům produktu IBM MQ , spíše než jedno z ostatních rozhraní API poskytovaných s produktem IBM MQ.
- Jakarta Messaging je nedílnou součástí Jakarta EE.

  Jakarta Messaging je přirozené rozhraní API, které se má použít pro zasílání zpráv na platformě Jakarta EE . Každý aplikační server, který je Jakarta EE vyhovující, musí obsahovat poskytovatele Jakarta Messaging . Produkt Jakarta Messaging můžete použít v aplikačních klientech, servletech, stránkách

Java Server Pages (JSP), objektech EJB (enterprise Java bean) a objektech typu message-driven bean (MDB). Všimněte si zejména, že aplikace Jakarta EE používají objekty MDB k asynchronnímu zpracování zpráv a všechny zprávy jsou doručeny do objektů MDB jako zprávy Jakarta Messaging .

- Továrny připojení a místa určení mohou být uloženy jako objekty spravované produktem Jakarta Messaging v centrálním úložišti, místo aby byly pevně naprogramovány do aplikace.

  Administrátor může vytvářet a udržovat spravované objekty Jakarta Messaging v centrálním úložišti a aplikace IBM MQ classes for Jakarta Messaging mohou tyto objekty načítat pomocí konzoly Java Naming Directory Interface (JNDI). Jakarta Messaging továrny připojení a místa určení zapouzdřují IBM MQspecifické informace, jako např. názvy správců front, názvy kanálů, volby připojení, názvy front a názvy témat. Pokud jsou továrny připojení a místa určení uložena jako spravované objekty, nejsou tyto informace pevně zakódovány do aplikace. Toto uspořádání proto poskytuje aplikaci určitý stupeň nezávislosti na základní konfiguraci produktu IBM MQ .
- Jakarta Messaging je rozhraní API odvětvového standardu, které může poskytovat přenositelnost aplikací.

  Aplikace Jakarta Messaging může používat produkt JNDI k načítání továren připojení a míst určení, která jsou uložena jako spravované objekty, a k provádění operací systému zpráv používat pouze rozhraní, která jsou definována v balíku `jakarta.jms` (Jakarta Messaging 3.0). Aplikace je pak zcela nezávislá na jakémkoli poskytovateli Jakarta Messaging , jako např. IBM MQ classes for Jakarta Messaging, a lze ji přenést z jednoho poskytovatele Jakarta Messaging do jiného bez jakékoli změny aplikace.

  Není-li produkt JNDI k dispozici v konkrétním prostředí aplikace, může aplikace IBM MQ classes for Jakarta Messaging použít rozšíření rozhraní API Jakarta Messaging k dynamickému vytváření a konfiguraci továren připojení a míst určení za běhu. Aplikace je pak zcela samostatná, ale je svázána s IBM MQ classes for Jakarta Messaging jako poskytovatel Jakarta Messaging .
- Pro aplikace mostu může být snazší psát pomocí produktu Jakarta Messaging.

  Aplikace mostu je aplikace, která přijímá zprávy z jednoho systému zpráv a odesílá je do jiného systému zpráv. Zápis aplikace mostu může být komplikován použitím rozhraní API a formátů zpráv specifických pro daný produkt. Namísto toho můžete napsat aplikaci mostu pomocí dvou poskytovatelů produktu Jakarta Messaging , jednoho pro každý systém zpráv. Aplikace poté použije pouze jedno rozhraní API, rozhraní API Jakarta Messaging a zpracuje pouze zprávy Jakarta Messaging .

Implementovatelná prostředí

Chcete-li zajistit integraci s aplikačním serverem Jakarta EE , standardy Jakarta EE vyžadují, aby poskytovatelé systému zpráv dodali adaptér prostředků. Podle specifikace Jakarta Connectors Architecture produkt IBM MQ poskytuje adaptér prostředků, který používá produkt Jakarta Messaging k poskytování funkcí systému zpráv v rámci jakéhokoli certifikovaného prostředí Jakarta EE . Další informace viz “Liberty a adaptér prostředků IBM MQ” na stránce 428.

Poznámka: WebSphere Application Server traditional v současné době nepodporuje Jakarta EE.

Mimo prostředí Jakarta EE jsou poskytnuty soubory OSGi a JAR, což usnadňuje získání pouze souboru IBM MQ classes for Jakarta Messaging. Tyto soubory JAR jsou snadněji implementovatelné buď samostatně, nebo v rámci rámců správy softwaru, například Maven. Další informace viz “Získání IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging samostatně” na stránce 122.

Související pojmy

IBM MQ třídy pro systém zpráv Jakarta: přehled

“Přístup k IBM MQ z Java -Volba rozhraní API” na stránce 82

Produkt IBM MQ poskytuje tři jazyková rozhraní Java .

JMS 2.0 Proč bych měl používat IBM MQ classes for JMS?

Použití produktu IBM MQ classes for JMS má řadu výhod, včetně možnosti znovu použít existující odbornosti produktu JMS ve vaší organizaci, a aplikací, které jsou více nezávislé na poskytovateli JMS a základní konfiguraci produktu IBM MQ .

Přehled výhod použití IBM MQ classes for JMS

Použití produktu IBM MQ classes for JMS vám umožňuje znovu použít existující odbornosti JMS a zajistit nezávislost aplikace.

V 9.3.0 V 9.3.0

Poznámka: JMS 2.0 byl nahrazen Jakarta Messaging. Produkt IBM MQ classes for JMS nadále podporuje standard JMS 2.0 , ale budoucí vylepšení systému zpráv Java se objeví pouze v souboru Jakarta Messaging, tedy v souboru IBM MQ classes for Jakarta Messaging. IBM MQ classes for JMS se doporučuje pouze pro údržbu a rozšíření existujících aplikací JMS 2.0 . IBM MQ classes for Jakarta Messaging by měla být upřednostňovanou technologií pro nový vývoj.

- Můžete znovu použít odbornosti JMS .

  IBM MQ classes for JMS je poskytovatel systému JMS , který implementuje rozhraní JMS pro systém zpráv IBM MQ . Pokud je vaše organizace v produktu IBM MQnová, ale již má schopnosti vývoje aplikací JMS , může být snazší používat známé rozhraní JMS API pro přístup k prostředkům IBM MQ , spíše než jedno z ostatních rozhraní API poskytovaných s produktem IBM MQ.
- JMS je nedílnou součástí Java Platform, Enterprise Edition (Java EE).

  JMS je přirozené rozhraní API, které se má použít pro zasílání zpráv na platformě Java EE . Každý aplikační server, který je Java EE vyhovující, musí obsahovat poskytovatele JMS . Produkt JMS můžete použít v aplikačních klientech, servletech, stránkách Java Server Pages (JSP), objektech EJB (enterprise Java bean) a objektech typu message-driven bean (MDB). Všimněte si zejména, že aplikace Java EE používají objekty MDB k asynchronnímu zpracování zpráv a všechny zprávy jsou doručeny do objektů MDB jako zprávy JMS .
- Továrny připojení a místa určení mohou být uloženy jako objekty spravované produktem JMS v centrálním úložišti, místo aby byly pevně naprogramovány do aplikace.

  Administrátor může vytvářet a udržovat spravované objekty JMS v centrálním úložišti a aplikace IBM MQ classes for JMS mohou tyto objekty načítat pomocí konzoly Java Naming Directory Interface (JNDI). JMS továrny připojení a místa určení zapouzdřují IBM MQspecifické informace, jako např. názvy správců front, názvy kanálů, volby připojení, názvy front a názvy témat. Pokud jsou továrny připojení a místa určení uložena jako spravované objekty, nejsou tyto informace pevně zakódovány do aplikace. Toto uspořádání proto poskytuje aplikaci určitý stupeň nezávislosti na základní konfiguraci produktu IBM MQ .
- JMS je rozhraní API odvětvového standardu, které může poskytovat přenositelnost aplikací.

  Aplikace JMS může použít produkt JNDI k načtení továren připojení a míst určení, která jsou uložena jako spravované objekty, a k provádění operací systému zpráv používat pouze rozhraní, která jsou definována v balíku `javax.jms` . Aplikace je pak zcela nezávislá na jakémkoli poskytovateli JMS , jako např. IBM MQ classes for JMS, a lze ji přenést z jednoho poskytovatele JMS do jiného bez jakékoli změny aplikace.

  Není-li produkt JNDI k dispozici v konkrétním prostředí aplikace, může aplikace IBM MQ classes for JMS používat rozšíření rozhraní API JMS k dynamickému vytváření a konfiguraci továren připojení a míst určení za běhu. Aplikace je pak zcela samostatná, ale je svázána s IBM MQ classes for JMS jako poskytovatel JMS .
- Pro aplikace mostu může být snazší psát pomocí produktu JMS.

  Aplikace mostu je aplikace, která přijímá zprávy z jednoho systému zpráv a odesílá je do jiného systému zpráv. Zápis aplikace mostu může být komplikován použitím rozhraní API a formátů zpráv specifických pro daný produkt. Namísto toho můžete napsat aplikaci mostu pomocí dvou poskytovatelů produktu JMS , jednoho pro každý systém zpráv. Aplikace poté použije pouze jedno rozhraní API, rozhraní API JMS a zpracuje pouze zprávy JMS .

Implementovatelná prostředí

Chcete-li zajistit integraci s aplikačním serverem Java EE , standardy Java EE vyžadují, aby poskytovatelé systému zpráv dodali adaptér prostředků. Podle specifikace Java EE Connector Architecture (JCA)

produkt IBM MQ poskytuje adaptér prostředků, který používá produkt JMS k poskytování funkcí systému zpráv v rámci jakéhokoli certifikovaného prostředí Java EE .

I když bylo možné použít IBM MQ classes for Java uvnitř Java EE, toto rozhraní API není navrženo nebo optimalizováno pro tento účel. Další informace o IBM MQ classes for Java aspektech v rámci Java EEviz “Spuštění aplikací IBM MQ classes for Java v rámci Java EE” na stránce 337.

Mimo prostředí Java EE jsou poskytnuty soubory OSGi a JAR, což usnadňuje získání pouze souboru IBM MQ classes for JMS. Tyto soubory JAR jsou snadněji implementovatelné buď samostatně, nebo v rámci rámců správy softwaru, například Maven. Další informace viz “Získání IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging samostatně” na stránce 122.

Související pojmy

V 9.3.0 V 9.3.0 IBM MQ třídy pro systém zpráv Jakarta: přehled

“Proč bych měl používat IBM MQ classes for Jakarta Messaging?” na stránce 79
Použití produktu IBM MQ classes for Jakarta Messaging má řadu výhod, včetně možnosti znovu použít existující odbornosti produktu Jakarta Messaging ve vaší organizaci, a aplikací, které jsou více nezávislé na poskytovateli Jakarta Messaging a základní konfiguraci produktu IBM MQ .

“Přístup k IBM MQ z Java -Volba rozhraní API” na stránce 82
Produkt IBM MQ poskytuje tři jazyková rozhraní Java .

Přístup k IBM MQ z Java -Volba rozhraní API

Produkt IBM MQ poskytuje tři jazyková rozhraní Java .

- V 9.3.0 V 9.3.0 IBM MQ classes for Jakarta Messaging
- IBM MQ classes for JMS
- IBM MQ classes for Java

IBM MQ classes for Jakarta Messaging

V 9.3.0 V 9.3.0 Produkt IBM MQ classes for Jakarta Messaging umožňuje aplikacím napsaným pomocí rozhraní Jakarta Messaging 3.0 API využívat produkt IBM MQ jako poskytovatele systému zpráv.

Jakarta Messaging je strategický směr pro zasílání zpráv v aplikacích Java .

Jakarta Messaging 3.0 je funkčně ekvivalentní s JMS 2.0, takže další informace viz “Použití IBM MQ classes for JMS/Jakarta Messaging” na stránce 78.

IBM MQ classes for JMS

Produkt IBM MQ classes for JMS umožňuje aplikacím napsaným pomocí rozhraní JMS 2.0 API využívat produkt IBM MQ jako poskytovatele systému zpráv.

Jako Jakarta Messaging nahrazující JMSse IBM MQ classes for JMS doporučuje používat v existujících aplikacích nebo v prostředích (například WebSphere Application Server), která nepodporují Jakarta Messaging.

Použití funkcí IBM MQ classes for Jakarta Messaging a IBM MQ classes for JMS ve stejné aplikaci není podporováno.

Další informace viz “Použití IBM MQ classes for JMS/Jakarta Messaging” na stránce 78.

IBM MQ classes for Java

Stabilized Další rozhraní API, které mohou aplikace Java používat pro přístup k IBM MQ prostředkům, je IBM MQ classes for Java, které poskytuje IBM MQorientované rozhraní API pro programy, které mohou používat IBM MQ jako poskytovatele systému zpráv. Produkt IBM MQ classes for Java je však funkčně stabilizován na úrovni dodané v produktu IBM MQ 8.0. Další informace viz “Proč bych měl používat IBM

MQ classes for Java?” na stránce 336. Ačkoli existující aplikace, které používají produkt IBM MQ classes for Java , jsou i nadále plně podporovány, nové aplikace by měly používat produkt IBM MQ classes for Jakarta Messaging.

Společné funkce IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging

IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging poskytují přístup jak k funkcím systému zpráv typu point-to-point, tak k funkcím systému zpráv typu publikování/odběr produktu IBM MQ. Kromě odesílání zpráv produktu JMS , které poskytují podporu pro standardní model systému zpráv JMS , mohou aplikace také odesílat a přijímat zprávy bez dalších záhlaví, a tak mohou spolupracovat s jinými aplikacemi IBM MQ , například s aplikacemi C MQI. K dispozici je úplné řízení informačního obsahu zpráv MQMD a MQ .

K dispozici jsou také další funkce produktu IBM MQ , jako např. streamování zpráv, asynchronní vložení a zprávy sestav.

Pomocí dodaných pomocných tříd PCF lze zprávy administrace produktu IBM MQ PCF odesílat a přijímat prostřednictvím rozhraní API produktu JMS a lze je použít k administraci správců front.

Funkce, které byly nedávno přidány do produktu IBM MQ, jako např. asynchronní spotřeba a automatické opětovné připojení, nejsou k dispozici v produktu IBM MQ classes for Java, ale jsou k dispozici v IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging.

Požadování vylepšení

Potřebujete-li přístup k funkcím produktu IBM MQ , které nejsou k dispozici prostřednictvím funkcí IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging, můžete nápad vyvolat.

Produkt IBM pak může poradit, zda je implementace možná v implementaci IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging , nebo zda existuje doporučený postup, který lze dodržet.

V případě dalších funkcí systému zpráv, protože produkt IBM je přispěvatelem k otevřenému standardu, mohou být tyto funkce zvýšeny jako součást procesu JCP. Ty by se vztahovaly pouze na zasílání zpráv Jakarta.

Související informace

Vítejte v portálu nápadů IBM

Proces přezkoumání specifikace JMS Java

Použití rozhraní JMS k odesílání zpráv PCF

V 9.3.0 JM 3.0 V 9.3.0 Předpoklady pro IBM MQ classes for Jakarta Messaging

Toto téma uvádí, co potřebujete vědět před použitím produktu IBM MQ classes for Jakarta Messaging. Chcete-li vyvíjet a spouštět aplikace IBM MQ classes for Jakarta Messaging , potřebujete určité softwarové komponenty jako nezbytné předpoklady.

Informace o předpokladech pro IBM MQ classes for Jakarta Messagingviz Systémové požadavky pro IBM MQ.

Chcete-li vyvíjet aplikace systému IBM MQ classes for Jakarta Messaging , potřebujete sadu Java SE Software Development Kit (SDK). Podrobné informace o JDK podporovaných vaším operačním systémem naleznete v části Systémové požadavky pro IBM MQ.

Chcete-li spustit aplikace IBM MQ classes for Jakarta Messaging , potřebujete následující softwarové komponenty:

- Správce front IBM MQ .
- Prostředí 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 z/OS, z/OS UNIX System Services (z/OS UNIX).

Poskytovatel JSSE IBM zahrnuje poskytovatele šifrování s certifikací FIPS, takže jej lze programově konfigurovat pro shodu FIPS 140-2 připravenou k okamžitému použití. Proto lze kompatibilitu se standardem FIPS 140-2 podporovat přímo z produktu IBM MQ classes for Jakarta Messaging.

Poskytovatel JSSE produktu Oraclemůže mít nakonfigurovaného poskytovatele šifrování s certifikací FIPS, ale není připraven k okamžitému použití a není k dispozici pro programovou konfiguraci. Proto v tomto případě produkt IBM MQ classes for Jakarta Messaging nemůže povolit shodu se standardem FIPS 140-2 přímo. Můžete být schopni ručně povolit tuto shodu, ale produkt IBM v tuto chvíli nemůže poskytnout vodítko.

Adresy Internet Protocol verze 6 (IPv6) můžete použít ve svých aplikacích IBM MQ classes for Jakarta Messaging , pokud jsou adresy IPv6 podporovány prostředím JVM (Java Virtual Machine) a implementací TCP/IP ve vašem operačním systému. Nástroj pro administraci IBM MQ Jakarta Messaging **JMS30Admin**také přijímá adresy IPv6 . Další informace o tomto nástroji naleznete v tématu Konfigurace objektů JMS a Jakarta Messaging pomocí administračních nástrojů.

Nástroj pro administraci IBM MQ JMS a IBM MQ Explorer používají Java Naming Directory Interface (JNDI) pro přístup k adresářové službě, která ukládá spravované objekty. Aplikace IBM MQ classes for Jakarta Messaging mohou také použít produkt JNDI k načtení spravovaných objektů z adresářové služby.

Poznámka: Pro Jakarta Messaging 3.0nemůžete spravovat rozhraní JNDI pomocí IBM MQ Explorer. Administrace rozhraní JNDI je podporována Jakarta Messaging 3.0 variantou **JMSAdmin**, což je **JMS30Admin**.

Poskytovatel služeb je kód, který poskytuje přístup k adresářové službě mapováním volání JNDI na adresářovou službu. Poskytovatel služeb systému souborů v souborech `fscontext.jar` a `providerutil.jar` je dodáván s produktem IBM MQ classes for Jakarta Messaging. Poskytovatel služeb systému souborů poskytuje přístup k adresářové službě na základě lokálního 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 tak, aby ukládal objekty Java . Informace o instalaci a konfiguraci serveru LDAP naleznete v dokumentaci dodané se serverem.

JMS 2.0 Předpoklady pro IBM MQ classes for JMS

Toto téma uvádí, co potřebujete vědět 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.

Informace o předpokladech pro IBM MQ classes for JMSviz Systémové požadavky pro IBM MQ.

Chcete-li vyvíjet aplikace systému IBM MQ classes for JMS , potřebujete sadu Java SE Software Development Kit (SDK). Podrobné informace o JDK podporovaných vaším operačním systémem naleznete v části Systémové požadavky pro IBM MQ.

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

- Správce front IBM MQ .
- Prostředí 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 z/OS, z/OS UNIX System Services (z/OS UNIX).

Poskytovatel JSSE IBM zahrnuje poskytovatele šifrování s certifikací FIPS, takže jej lze programově konfigurovat pro shodu FIPS 140-2 připravenou k okamžitému použití. Proto lze kompatibilitu se standardem FIPS 140-2 podporovat přímo z IBM MQ classes for Java a IBM MQ classes for JMS.

Poskytovatel JSSE produktu Oraclemůže mít nakonfigurovaného poskytovatele šifrování s certifikací FIPS, 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 přímo povolit shodu se

standardem FIPS 140-2. Můžete být schopni ručně povolit tuto shodu, ale produkt IBM v tuto chvíli nemůže poskytnout vodítko.

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

Nástroj pro administraci IBM MQ JMS a IBM MQ Explorer používají Java Naming Directory Interface (JNDI) pro přístup k adresářové službě, která ukládá spravované objekty. Aplikace IBM MQ classes for JMS mohou také použít produkt JNDI k načtení spravovaných objektů z adresářové služby. Poskytovatel služeb je kód, který poskytuje přístup k adresářové službě mapováním volání JNDI na adresářovou službu. Poskytovatel služeb 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 na základě lokálního 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 tak, aby ukládal objekty Java . Informace o instalaci a konfiguraci serveru LDAP naleznete v dokumentaci dodané se serverem.

Instalace a konfigurace IBM MQ classes for JMS/Jakarta Messaging

Tento oddíl popisuje adresáře a soubory, které jsou vytvořeny při instalaci produktu IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging, a uvádí, jak nakonfigurovat IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging po instalaci.

Související pojmy

“Použití adaptéru prostředků IBM MQ” na stránce 423

Adaptér prostředků umožňuje aplikacím spuštěným na aplikačním serveru přistupovat k prostředkům IBM MQ . Podporuje příchozí a odchozí komunikaci.

Co je instalováno pro produkt IBM MQ classes for JMS

Při instalaci produktu IBM MQ classes for JMSse vytvoří několik souborů a adresářů. V systému Windowsse některá konfigurace provádí během instalace automatickým nastavením proměnných prostředí. Na jiných platformách a v určitých prostředích Windows musíte před spuštěním aplikací IBM MQ classes for JMS nastavit proměnné prostředí.

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

Další informace o instalaci produktu IBM MQviz:

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

Důležité: Kromě přemístitelných souborů JAR popsaných v tématu “IBM MQ classes for JMS/Jakarta Messaging přemístitelné soubory JAR” na stránce 87není podporováno kopírování souborů JAR IBM MQ classes for JMS nebo nativních knihoven do jiných počítačů nebo do jiného umístění na počítači, kde byl nainstalován produkt IBM MQ classes for JMS .

Instalační adresáře

Tabulka 5 na stránce 86 ukazuje, kde jsou na každé platformě nainstalovány soubory IBM MQ classes for JMS .

Tabulka 5. IBM MQ classes for JMS instalační adresáře	
Platforma	**Adresář**
Linux AIX AIX 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*/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 , včetně přemístitelných souborů JAR, které se nacházejí v adresáři *MQ_INSTALLATION_PATH*\java\lib .
- Nativní knihovny IBM MQ používané aplikacemi, které používají nativní rozhraní Java .

  32bitové nativní knihovny se instalují do adresáře *MQ_INSTALLATION_PATH*\java\lib a 64bitové nativní knihovny se nacházejí v adresáři *MQ_INSTALLATION_PATH*\java\lib64 .

  Další informace o nativních knihovnách IBM MQ viz “Konfigurace knihoven JNI (Java Native Interface)” na stránce 92.
- Další skripty, které jsou popsány v části “Skripty dodávané s IBM MQ classes for JMS/Jakarta Messaging” na stránce 119. Tyto skripty se nacházejí v adresáři *MQ_INSTALLATION_PATH*\java\bin .
- Specifikace rozhraní API IBM MQ classes for JMS . Nástroj Javadoc byl použit ke generování stránek HTML, které obsahují specifikace rozhraní API.

  Stránky HTML jsou v adresáři *MQ_INSTALLATION_PATH*\java\doc\WMQJMSClasses :

  – ALW V systému AIX, Linux, and Windowstento podadresář obsahuje jednotlivé stránky HTML.

  – IBM i V systému IBM ijsou stránky HTML v souboru s názvem wmqjms_javadoc.jar.

  – z/OS V systému z/OSjsou stránky HTML v souboru s názvem wmqjms_javadoc.jar.
- Podpora pro OGSi. Balíky OSGi jsou nainstalovány v adresáři java\lib\OSGi a popsány v souboru “Podpora pro OSGi s IBM MQ classes for JMS” na stránce 120.
- Adaptér prostředků IBM MQ , který lze implementovat do libovolného aplikačního serveru kompatibilního s Java Platform, Enterprise Edition 7 (Java EE 7) nebo Jakarta EE .

  Adaptér prostředků IBM MQ se nachází v adresáři *MQ_INSTALLATION_PATH*\java\lib\jca ; další informace viz “Použití adaptéru prostředků IBM MQ” na stránce 423
- Windows V systému Windowsjsou symboly, které lze použít pro ladění, instalovány v adresáři *MQ_INSTALLATION_PATH*\java\lib\symbols .

Instalační adresář také obsahuje některé soubory, které patří do jiných komponent produktu IBM MQ .

Ukázkové aplikace

JMS 2.0 Některé ukázkové aplikace jsou dodávány s produktem IBM MQ classes for JMS. Tabulka 6 na stránce 86 ukazuje, kde jsou na každé platformě nainstalovány ukázkové aplikace.

JM 3.0 Pro IBM MQ classes for Jakarta Messagingse připravují nové ukázky.

JMS 2.0

Tabulka 6. Adresáře ukázek pro IBM MQ classes for JMS	
Platforma	**Adresář**
Linux AIX AIX 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*/java/samples/jms

V této tabulce představuje *MQ_INSTALLATION_PATH* adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Po instalaci budete možná muset provést některé konfigurační úlohy pro kompilaci a spuštění aplikací.

"Nastavení proměnných prostředí pro IBM MQ classes for JMS/Jakarta Messaging" na stránce 90 popisuje cestu ke třídě, která je nezbytná pro spuštění ukázkových IBM MQ classes for JMS aplikací. Toto téma také popisuje další soubory JAR, na které je třeba odkazovat za zvláštních okolností, a proměnné prostředí, které musíte nastavit tak, aby spouštěli skripty dodávané s produktem IBM MQ classes for JMS.

Chcete-li řídit vlastnosti, jako je trasování a protokolování aplikace, musíte poskytnout soubor vlastností konfigurace. Soubor vlastností konfigurace IBM MQ classes for JMS je popsán v části "Konfigurační soubor IBM MQ classes for JMS/Jakarta Messaging" na stránce 95.

Související pojmy

Problémy při implementaci adaptéru prostředků

Související úlohy

"Použití ukázkových aplikací IBM MQ classes for JMS" na stránce 115
Ukázkové aplikace IBM MQ classes for JMS poskytují přehled obecných funkcí rozhraní API produktu JMS . Můžete je použít k ověření instalace a nastavení serveru systému zpráv a k usnadnění vytváření vlastních aplikací.

IBM MQ classes for JMS/Jakarta Messaging přemístitelné soubory JAR
Přemístitelné soubory JAR lze přesunout na systémy, které potřebují spustit IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging.

Důležité:

- Kromě relokovatelných souborů JAR popsaných v části Relokovatelné soubory JARnení podporováno kopírování souborů JAR IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging nebo nativních knihoven do jiných počítačů nebo do jiného umístění na počítači, kde byly nainstalovány soubory JAR IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging .
- Nezahrnujte přemístitelné soubory JAR do aplikací implementovaných do aplikačních serverů Java EE , jako např. WebSphere Application Server nebo WebSphere Liberty. V těchto prostředích by měl být implementován a použit adaptér prostředků IBM MQ . Všimněte si, že produkt WebSphere Application Server vkládá adaptér prostředků IBM MQ , takže jej není nutné ručně implementovat do tohoto prostředí.
- Chcete-li se vyhnout konfliktům zavaděče tříd, nedoporučuje se zabalit přemístitelné soubory JAR do více aplikací ve stejném běhovém prostředí Java . V tomto scénáři zpřístupněte přemístitelné soubory JAR IBM MQ na cestě ke třídě běhového prostředí Java .
- Pokud ve svých aplikacích sdružujete přemístitelné soubory JAR, ujistěte se, že jste zahrnuli všechny nezbytné soubory JAR, jak je popsáno v tématu Přemísťitelné soubory JAR. Měli byste se také ujistit, že máte příslušné procedury pro aktualizaci souborů JAR v balíku v rámci údržby aplikace, abyste se ujistili, že IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging jsou stále aktuální a známé problémy jsou znovu mediovány.

Přemístitelné soubory JAR

V rámci podniku lze následující soubory přesunout na systémy, na kterých je třeba spustit příkaz IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging:

- V 9.3.0 V 9.3.0 `bcpkix-jdk15to18.jar` "4" na stránce 88
- V 9.3.5 `bcpkix-jdk18on.jar` "3" na stránce 88
- V 9.3.0 V 9.3.0 `bcprov-jdk15to18.jar` "4" na stránce 88
- V 9.3.5 `bcprov-jdk18on.jar` "3" na stránce 88
- V 9.3.0 V 9.3.0 `bcutil-jdk15to18.jar` "4" na stránce 88
- V 9.3.5 `bcutil-jdk18on.jar` "3" na stránce 88
- JMS 2.0 `com.ibm.mq.allclient.jar` "1" na stránce 88
- V 9.3.0 JM 3.0 V 9.3.0 `com.ibm.mq.jakarta.client.jar` "2" na stránce 88
- V 9.3.3 Removed `com.ibm.mq.traceControl.jar`
- `fscontext.jar`
- V 9.3.3 `jackson-annotations.jar`
- V 9.3.3 `jackson-core.jar`
- V 9.3.3 `jackson-databind.jar`
- `jakarta.jms-api.jar`
- `jms.jar`
- `org.json.jar`
- `providerutil.jar`

Notes:

1. JMS 2.0 a JMS 1.1
2. Jakarta Messaging 3.0
3. Continuous DeliveryOdIBM MQ 9.3.5
4. Long Term Support a Continuous Delivery před IBM MQ 9.3.5

JMS souborů JAR

V 9.3.0 V 9.3.0 `jms.jar` obsahuje rozhraní JMS 1.1 a JMS 2.0 -tato se nazývají `javax.jms.*`.

V 9.3.0 JM 3.0 V 9.3.0 `jakarta.jms-api.jar` obsahuje rozhraní Jakarta Messaging 3.0 -tato rozhraní jsou pojmenována `jakarta.jms.*`.

fscontext.jar a providerutil.jar

Soubory `fscontext.jar` a `providerutil.jar` jsou nezbytné, pokud vaše aplikace provádí vyhledávání rozhraní JNDI pomocí kontextu systému souborů.

Poskytovatel zabezpečení Bouncy Castle a CMS podporují soubory JAR

Poskytovatel zabezpečení Bouncy Castle a soubory JAR podpory CMS jsou povinné. Další informace viz Podpora jiných prostředí JRE nežIBM s AMS.

V 9.3.5 Pro Continuous Delivery z IBM MQ 9.3.5jsou nezbytné následující soubory JAR:

- `bcpkix-jdk18on.jar`
- `bcprov-jdk18on.jar`
- `bcutil-jdk18on.jar`

LTS V případě souborů Long Term Support a Continuous Delivery před IBM MQ 9.3.5jsou vyžadovány následující soubory JAR:

- `bcpkix-jdk15to18.jar`
- `bcprov-jdk15to18.jar`
- `bcutil-jdk15to18.jar`

org.json.jar

Soubor `org.json.jar` je povinný, pokud vaše aplikace IBM MQ classes for JMS používá CCDT ve formátu JSON.

com.ibm.mq.allclient.jar a com.ibm.mq.jakarta.client.jar

Soubory `com.ibm.mq.allclient.jar` a `com.ibm.mq.jakarta.client.jar` obsahují třídy IBM MQ classes for JMS, IBM MQ classes for Jakarta Messaging, IBM MQ classes for Javaa PCF a záhlaví. Pokud přesunete tento soubor JAR do nového umístění, ujistěte se, že jste provedli kroky k udržení tohoto nového umístění s novými opravnými sadami IBM MQ Fix Pack. Také se ujistěte, že použití souborů je známo podpoře IBM , pokud dostáváte prozatímní opravu.

Chcete-li určit verzi souborů `com.ibm.mq.allclient.jar` a `com.ibm.mq.jakarta.client.jar`, použijte následující příkaz:

V 9.3.0 JM 3.0 V 9.3.0

```
java -jar com.ibm.mq.jakarta.client.jar
```

JMS 2.0

```
java -jar com.ibm.mq.allclient.jar
```

Následující příklad ukazuje ukázku výstupu z tohoto příkazu:

```
C:\Program Files\IBM\MQ_1\java\lib>java -jar com.ibm.mq.allclient.jar
Name:       Java Message Service Client
Version:    9.3.0.0
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:    9.3.0.0
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:    9.3.0.0
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:    9.3.0.0
Level:      p000-L140428.1
Build Type: Production
Location:   file:/C:/Program Files/IBM/MQ_1/java/lib/com.ibm.mq.allclient.jar
```

jackson-annotations.jar, jackson-core.jar a jackson-databind.jar

V 9.3.3

Pokud vaše aplikace IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging vytvoří zabezpečená připojení TLS ke správci front, jsou vyžadovány tři soubory JAR Jackson.

Nastavení proměnných prostředí pro IBM MQ classes for JMS/Jakarta Messaging
Než budete moci kompilovat a spouštět aplikace IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging , musí nastavení pro proměnnou prostředí **CLASSPATH** obsahovat soubor JAR (IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging Java archive). V závislosti na vašich požadavcích budete možná muset do cesty ke třídám přidat další soubory JAR. Chcete-li spustit skripty dodávané s produktem IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging, musí být nastaveny další proměnné prostředí.

Než začnete

V 9.3.0 JM 3.0 V 9.3.0 Od IBM MQ 9.3.0, Jakarta Messaging 3.0 je podporován pro vývoj nových aplikací. Produkt IBM MQ 9.3.0 nadále podporuje produkt JMS 2.0 pro existující aplikace. Použití rozhraní API Jakarta Messaging 3.0 a rozhraní API JMS 2.0 ve stejné aplikaci není podporováno. Další informace naleznete v tématu Použití tříd IBM MQ pro systém zpráv JMS/Jakarta.

Důležité: Nastavení volby Java *-Xbootclasspath* tak, aby zahrnovala IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging , není podporováno.

Informace o této úloze

Chcete-li kompilovat a spouštět aplikace IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging , použijte nastavení **CLASSPATH** pro vaši platformu a verzi systému zpráv Java , jak ukazuje následující tabulka. Případně můžete zadat cestu ke třídám v příkazu **java** namísto použití proměnné prostředí.

JMS 2.0 Pro systém IBM MQ classes for JMSnastavení zahrnuje adresář ukázek, takže můžete zkompilovat a spustit ukázkové aplikace IBM MQ classes for JMS .

JM 3.0 Pro IBM MQ classes for Jakarta Messagingse připravují nové ukázky.

JM 3.0

*Tabulka 7. **CLASSPATH** nastavení pro Jakarta Messaging 3.0 ke kompilaci a spuštění aplikací IBM MQ classes for Jakarta Messaging*

Platforma	CLASSPATH nastavení
AIX AIX	CLASSPATH= *MQ_INSTALLATION_PATH*/java/lib/com.ibm.mq.jakarta.client.jar:
Linux Linux	CLASSPATH= *MQ_INSTALLATION_PATH*/java/lib/com.ibm.mq.jakarta.client.jar:
IBM i IBM i	CLASSPATH=/QIBM/ProdData/mqm/java/lib/com.ibm.mq.jakarta.client.jar:
Windows Windows	CLASSPATH= *MQ_INSTALLATION_PATH*\java\lib\com.ibm.mq.jakarta.client.jar;
z/OS z/OS	CLASSPATH= *MQ_INSTALLATION_PATH*/java/lib/com.ibm.mq.jakarta.client.jar;

JMS 2.0

*Tabulka 8. **CLASSPATH** nastavení pro produkt JMS 2.0 ke kompilaci a spuštění aplikací IBM MQ classes for JMS , včetně ukázkových aplikací*	
Platforma	**Nastavení CLASSPATH**
AIX	CLASSPATH= *MQ_INSTALLATION_PATH*/java/lib/com.ibm.mq.allclient.jar: *MQ_INSTALLATION_PATH*/samp/jms/samples:
Linux	CLASSPATH= *MQ_INSTALLATION_PATH*/java/lib/com.ibm.mq.allclient.jar: *MQ_INSTALLATION_PATH*/samp/jms/samples:
IBM i	CLASSPATH=/QIBM/ProdData/mqm/java/lib/com.ibm.mq.allclient.jar: /QIBM/ProdData/mqm/java/samples/jms/samples:
Windows	CLASSPATH= *MQ_INSTALLATION_PATH*\java\lib\com.ibm.mq.allclient.jar; *MQ_INSTALLATION_PATH*\tools\jms\samples;
z/OS	CLASSPATH= *MQ_INSTALLATION_PATH*/java/lib/com.ibm.mq.allclient.jar: *MQ_INSTALLATION_PATH*/java/samples/jms/samples:

V těchto tabulkách představuje *MQ_INSTALLATION_PATH* adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Soubor typu manifest souboru JAR `com.ibm.mq.jakarta.client.jar` nebo `com.ibm.mq.allclient.jar` obsahuje odkazy na většinu ostatních souborů JAR vyžadovaných aplikacemi IBM MQ classes for JMS , a proto tyto soubory JAR nemusíte přidávat do své cesty ke třídám. Tyto soubory JAR zahrnují soubory vyžadované aplikacemi, které používají rozhraní JNDI (Java Naming Directory Interface) k načtení spravovaných objektů z adresářové služby, a aplikacemi, které používají rozhraní JTA (Java Transaction API).

Do cesty ke třídám však musíte zahrnout další soubory JAR za následujících okolností:

- Používáte-li třídy uživatelské procedury kanálu, které implementují rozhraní uživatelské procedury kanálu definovaná v balíku `com.ibm.mq` , musíte místo těch, které jsou definovány v balíku `com.ibm.mq.exits` , přidat do své cesty ke třídám soubor IBM MQ classes for Java JAR `com.ibm.mq.jar`.
- Pokud vaše aplikace používá rozhraní JNDI k načtení spravovaných objektů z adresářové služby, musíte také přidat následující soubory JAR do cesty ke třídám:
  - `fscontext.jar`
  - `providerutil.jar`
- Pokud vaše aplikace používá JTA, musíte také přidat `jta.jar` do cesty ke třídám.

Poznámka: Tyto další soubory JAR jsou nezbytné pouze pro kompilaci aplikací, nikoli pro jejich spuštění.

Skripty poskytované s produktem IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging používají následující proměnné prostředí:

MQ_JAVA_DATA_PATH
Tato proměnná prostředí určuje adresář pro výstup protokolu a trasování.

MQ_JAVA_INSTALL_PATH
Tato proměnná prostředí určuje adresář, kde je nainstalován produkt IBM MQ classes for JMS .

MQ_JAVA_LIB_PATH
Tato proměnná prostředí určuje adresář, kde jsou uloženy knihovny IBM MQ classes for JMS , jak je uvedeno v předchozích tabulkách.

Procedura

- Windows

  V systému Windowspo instalaci produktu IBM MQspusťte příkaz **setmqenv**.

  Pokud tento příkaz nespustíte jako první, může se při zadávání příkazu **dspmqver** zobrazit následující chybová zpráva:

  ```
  AMQ8351: IBM MQ Prostředí Java nebylo nakonfigurováno
  nebo funkce IBM MQ JRE nebyla nainstalována.
  ```

  Poznámka: Tato zpráva se očekává, pokud jste nenainstalovali prostředí IBM MQ Java runtime environment (JRE) (viz Kontrola předpokladů dalších funkcí systému Windows.

- Linux AIX

  Na systémech AIX and Linux nastavte proměnné prostředí sami:

  JMS 2.0 Pro systém JMS 2.0použijte jeden z následujících skriptů k nastavení proměnných prostředí:

  – Používáte-li 32bitové prostředí JVM, použijte skript `setjmsenv`.
  – Používáte-li 64bitové prostředí JVM na systému AIX nebo Linux , použijte skript `setjmsenv64`.

  JM 3.0 Pro systém Jakarta Messaging 3.0použijte jeden z následujících skriptů k nastavení proměnných prostředí:

  – Používáte-li 32bitové prostředí JVM, použijte skript `setjms30env`.
  – Používáte-li 64bitové prostředí JVM, použijte skript `setjms30env64`.

  Tyto skripty jsou v adresáři *MQ_INSTALLATION_PATH*`/java/bin` , kde *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

  Tyto skripty můžete použít různými způsoby. Skript můžete použít jako základ pro nastavení požadovaných proměnných prostředí, jak je uvedeno v tabulce, nebo je můžete přidat do souboru `.profile` pomocí textového editoru. Máte-li netypické nastavení, upravte obsah skriptu podle potřeby. Případně můžete spustit skript v každé relaci, ze které se mají spustit spouštěcí skripty JMS . Pokud vyberete tuto volbu, musíte spustit skript v každém okně shellu, které spustíte, během procesu ověření JMS :

  – JMS 2.0 Jako parametr JMS 2.0zadejte `. ./setjmsenv` nebo `../setjmsenv64`.
  – JM 3.0 Jako parametr Jakarta Messaging 3.0zadejte `. ./setjms30env` nebo `. ./setjms30env64`.

  IBM i V systému IBM imusíte nastavit proměnnou prostředí **QIBM_MULTI_THREADED** na hodnotu Y. Poté můžete spustit aplikace s podporou podprocesů stejným způsobem, jakým spouštíte aplikace s podporou podprocesů. Další informace naleznete v tématu Nastavení produktu IBM MQ s produkty Java a JMS.

Související úlohy

“Použití ukázkových aplikací IBM MQ classes for JMS” na stránce 115
Ukázkové aplikace IBM MQ classes for JMS poskytují přehled obecných funkcí rozhraní API produktu JMS . Můžete je použít k ověření instalace a nastavení serveru systému zpráv a k usnadnění vytváření vlastních aplikací.

Související odkazy

“Skripty dodávané s IBM MQ classes for JMS/Jakarta Messaging” na stránce 119
K dispozici je řada skriptů, které pomáhají s běžnými úlohami, které je třeba provést při použití produktu IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging.

Konfigurace knihoven JNI (Java Native Interface)
Aplikace systému IBM MQ classes for JMS , které se buď připojují ke správci front pomocí transportu vazeb, nebo se připojují ke správci front pomocí transportu klienta a používají programy uživatelské

procedury kanálu napsané v jiných jazycích než Java, musí být spuštěny v prostředí, které umožňuje přístup ke knihovnám Java Native Interface (JNI).

Než začnete

Další informace o použití prostředí WebSphere Application Server naleznete v tématu Konfigurace poskytovatele systému zpráv IBM MQ s informacemi o nativních knihovnách .

Informace o této úloze

Chcete-li nastavit toto prostředí, musíte nakonfigurovat cestu ke knihovně prostředí tak, aby prostředí Java Virtual Machine (JVM) mohlo před spuštěním aplikace IBM MQ classes for JMS načíst knihovnu mqjbnd.

Produkt IBM MQ poskytuje dvě Java knihovny nativního rozhraní (JNI):

mqjbnd
Tuto knihovnu používají aplikace, které se připojují ke správci front pomocí přenosu vazeb. Poskytuje rozhraní mezi produktem IBM MQ classes for JMS a správcem front. Knihovnu mqjbnd nainstalovanou s produktem IBM MQ 9.3 lze použít pro připojení k libovolnému správci front IBM MQ 9.3 (nebo dřívějšímu).

mqjexitstub02
Knihovna mqjexitstub02 je načtena produktem IBM MQ classes for JMS , když se aplikace připojuje ke správci front pomocí přenosu klienta a používá program uživatelské procedury kanálu napsaný v jiném jazyce než Java.

Na určitých platformách produkt IBM MQ instaluje 32bitové a 64bitové verze těchto knihoven JNI. Umístění knihoven pro každou platformu je uvedeno v Tabulce 1.

Tabulka 9. Umístění knihoven IBM MQ classes for JMS pro každou platformu

Platforma	**Adresář obsahující knihovny IBM MQ classes for JMS**
AIX AIX Linux Linux (platformy POWER, x86-64 a zSeries s390x)	*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 (64bitové knihovny)
z/OS z/OS	*MQ_INSTALLATION_PATH*/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 31bitové, nebo 64bitové prostředí Java Virtual Machine (JVM). Není nutné určovat, které knihovny JNI mají být použity. Produkt IBM MQ classes for JMS může sám pro sebe určit, které knihovny JNI mají být načítány.

Postup

1. Nakonfigurujte vlastnost prostředí JVM **`java.library.path`** , kterou lze provést dvěma způsoby:
   - Zadáním argumentu prostředí JVM, jak je uvedeno 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 nastavilo svůj vlastní soubor `java.library.path`. Tato cesta se liší podle platformy a umístění, ve kterém jste nainstalovali produkt IBM MQ. Například pro 64bitové prostředí JVM a výchozí umístění instalace produktu IBM MQ můžete použít následující nastavení:

  AIX
  ```
  export LIBPATH=/usr/mqm/java/lib64:$LIBPATH
  ```

  Linux
  ```
  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, když prostředí nebylo správně nakonfigurováno, je následující:

```
Příčina: com.ibm.mq.jmqi.local.LocalMQ$4: CC=2;RC=2495;
AMQ8598: Nepodařilo se načíst nativní knihovnu JNI produktu WebSphere MQ : 'mqjbnd'.
    v com.ibm.mq.jmqi.local.LocalMQ.loadLib(LocalMQ.java:1268)
    na com.ibm.mq.jmqi.local.LocalMQ$1.run(LocalMQ.java:309)
    v java.security.AccessController.doPrivileged(AccessController.java:400)
    na 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 com.ibm.mq.jmqi.local.LocalServer. < init> (LocalServer.java:230)
    v sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native metoda)

v sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:86)

    na adrese
sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.jav
a:58)
    v java.lang.reflect.Constructor.newInstance(Constructor.java:542)
    v adresáři com.ibm.mq.jmqi.JmqiEnvironment.getInstance(JmqiEnvironment.java:706)
    v adresáři com.ibm.mq.jmqi.JmqiEnvironment.getMQI(JmqiEnvironment.java:640)

v com.ibm.msg.client.wmq.factories.WMQConnectionFactory.createV7ProviderConnection(WMQConnectio
nFactory.java:8437)
    ... 7 dalších
Příčina: java.lang.UnsatisfiedLinkError: mqjbnd (nenalezeno v java.library.path)
    na java.lang.ClassLoader.loadLibraryWithPath(ClassLoader.java:1235)
    v java.lang.ClassLoader.loadLibraryWithClassLoader(ClassLoader.java:1205)
    v java.lang.System.loadLibrary(System.java:534)
    v com.ibm.mq.jmqi.local.LocalMQ.loadLib(LocalMQ.java:1240)
    ... 20 dalších
```

2. Po nastavení 32bitového nebo 64bitového prostředí spusťte aplikaci IBM MQ classes for JMS pomocí příkazu:

   ```
   java application-name
   ```

   kde *název_aplikace* je název aplikace IBM MQ classes for JMS , která má být spuštěna.

   Výjimka obsahující IBM MQ Kód příčiny 2495 (MQRC_MODULE_NOT_FOUND) je vyvolána IBM MQ classes for JMS , pokud:

   - Aplikace IBM MQ classes for JMS je spuštěna ve 32bitovém prostředí Java runtime environmenta pro server IBM MQ classes for JMSbylo nastaveno 64bitové prostředí, protože 32bitový produkt Java runtime environment nemůže načíst 64bitovou nativní knihovnu Java .
   - Aplikace IBM MQ classes for JMS je spuštěna v 64bitovém prostředí Java runtime environmenta pro server 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/Jakarta Messaging
Konfigurační soubory IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging uvádějí vlastnosti, které se používají ke konfiguraci IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging.

Poznámka: Vlastnosti definované v konfiguračním souboru lze také nastavit jako systémové vlastnosti prostředí JVM. Je-li vlastnost nastavena jak v konfiguračním souboru, tak jako systémová vlastnost, má přednost systémová vlastnost. Proto můžete v případě potřeby přepsat libovolnou vlastnost v konfiguračním souboru tak, že ji zadáte jako systémovou vlastnost v příkazu **java** .

Formát konfiguračního souboru IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging je formát standardního souboru vlastností Java . Ukázkový konfigurační soubor s názvem `jms.config` se dodává v podadresáři `bin` instalačního adresáře 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 nebo IBM MQ classes for Jakarta Messaging . Při spouštění aplikace použijte příkaz **java** v následujícím formátu:

```
java -Dcom.ibm.msg.client.config.location= config_file_url application_name
```

V příkazu *config_file_url* je jednotný lokátor prostředků (URL), který určuje název a umístění konfiguračního souboru IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging . Jsou podporovány adresy URL následujících typů: http, file, ftp a 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 nebo IBM MQ classes for Jakarta Messaging jako soubor `D:\mydir\mjms.config` na lokálním systému Windows .

Při spuštění aplikace produkt IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging přečte obsah konfiguračního souboru a uloží zadané vlastnosti do interního úložiště vlastností. Pokud příkaz **java** neidentifikuje konfigurační soubor, nebo pokud konfigurační soubor nelze nalézt, IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging použije výchozí hodnoty pro všechny vlastnosti.

Konfigurační soubor IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging lze použít s jakýmkoli podporovaným přenosem mezi aplikací a správcem front nebo zprostředkovatelem.

Potlačení vlastností uvedených 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 nebo IBM MQ classes for Jakarta Messaging. Vlastnosti zadané v konfiguračním souboru IBM MQ MQI client se však použijí pouze v případě, že se aplikace připojí ke správci front v režimu klienta.

V případě potřeby můžete přepsat libovolný atribut v konfiguračním souboru IBM MQ MQI client tak, že jej zadáte jako vlastnost v konfiguračním souboru IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging . Chcete-li přepsat atribut v konfiguračním souboru IBM MQ MQI client , použijte položku s následujícím formátem v konfiguračním souboru IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging :

```
com.ibm.mq.cfg. stanza. propName = propValue
```

Proměnné v položce mají následující význam:

sekce
Název sekce 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 uvedenou v konfiguračním souboru IBM MQ MQI client .

Případně můžete přepsat atribut v konfiguračním souboru IBM MQ MQI client zadáním vlastnosti jako systémové vlastnosti v příkazu **java** . K určení vlastnosti jako systémové vlastnosti použijte předchozí formát.

Pouze následující atributy v konfiguračním souboru IBM MQ MQI client jsou důležité pro IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging. Zadáte-li nebo potlačíte-li jiné atributy, nebude to mít žádný účinek. Konkrétně si uvědomte, že `ChannelDefinitionFile` a `ChannelDefinitionDirectory` v sekci CHANNELS konfiguračního souboru klienta se nepoužívají. Podrobnosti o použití tabulky CCDT s IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messagingviz “Použití tabulky definic kanálů klienta s volbou IBM MQ classes for JMS” na stránce 275 .

Tabulka 10. Která sekce konfiguračního souboru klienta obsahuje který atribut

sekce	**Atribut**
CHANNELS stanza konfiguračního souboru klienta	Put1DefaultAlwaysSync
CHANNELS stanza konfiguračního souboru klienta	DefRecon
CHANNELS stanza konfiguračního souboru klienta	ReconDelay
CHANNELS stanza konfiguračního souboru klienta	PasswordProtection
ClientExitSekce cesty konfiguračního souboru klienta	ExitsDefaultPath
ClientExitSekce cesty konfiguračního souboru klienta	ExitsDefaultPath64
ClientExitSekce cesty konfiguračního souboru klienta	JavaExitsClasspath
Sekce JMQI konfiguračního souboru klienta	useMQCSPauthentication
MessageBuffer sekce konfiguračního souboru klienta	MaximumSize
MessageBuffer sekce konfiguračního souboru klienta	PurgeTime
MessageBuffer sekce konfiguračního souboru klienta	UpdatePercentage
stanza TCP konfiguračního souboru klienta	ClntRcvBufSize
stanza TCP konfiguračního souboru klienta	ClntSndBufSize
stanza TCP konfiguračního souboru klienta	Vypršení časového limitu připojení
stanza TCP konfiguračního souboru klienta	KeepAlive

Další podrobnosti o konfiguraci IBM MQ MQI client viz konfigurační soubor IBM MQ MQI client , `mqclient.ini`

Použití Java standardního trasování prostředí ke konfiguraci trasování JMS
Pomocí sekce Java Standardní nastavení trasování prostředí nakonfigurujte trasovací prostředek IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging .

com.ibm.msg.client.commonservices.trace.outputName = *traceOutputNázev*
`traceOutputName` je název adresáře a souboru, do kterého se odesílá výstup trasování.

Standardně se informace o trasování zapisují do trasovacího souboru v aktuálním pracovním adresáři aplikace. Název trasovacího souboru závisí na prostředí, ve kterém je aplikace spuštěna:

- V 9.3.0 JM 3.0 V 9.3.0 V systému IBM MQ 9.3.0, pokud aplikace načetla soubor IBM MQ classes for Jakarta Messaging z přemístitelného souboru JAR `com.ibm.mq.jakarta.client.jar` (Jakarta Messaging 3.0) nebo IBM MQ classes for JMS z přemístitelného souboru JAR `com.ibm.mq.allclient.jar` (JMS 2.0), je trasování zapsáno do souboru s názvem `mqjavaclient_%PID%.cl%u.trc`.
- Z adresáře IBM MQ 9.1.5 a IBM MQ 9.1.0 Fix Pack 5:
  - Pokud aplikace načetla soubor IBM MQ classes for JMS ze souboru JAR, který lze přemístit `com.ibm.mq.allclient.jar`, trasování se zapíše do souboru s názvem `mqjavaclient_%PID%.cl%u.trc`.
  - Pokud aplikace načetla soubor IBM MQ classes for JMS ze souboru JAR `com.ibm.mqjms.jar`, trasování se zapíše do souboru s názvem `mqjava_%PID%.cl%u.trc`.
- Z adresáře IBM MQ 9.0.0 Fix Pack 2:
  - Pokud aplikace načetla soubor IBM MQ classes for JMS ze souboru JAR, který lze přemístit `com.ibm.mq.allclient.jar`, trasování se zapíše do souboru s názvem `mqjavaclient_%PID%.trc`.
  - Pokud aplikace načetla soubor IBM MQ classes for JMS ze souboru JAR `com.ibm.mqjms.jar`, trasování se zapíše do souboru s názvem `mqjava_%PID%.trc`.
- V případě systému IBM MQ classes for JMS for IBM MQ 9.0.0 Fix Pack 1 nebo dřívějšího se trasování zapisuje do souboru s názvem `mqjms_%PID%.trc`.

kde *%PID%* je identifikátor procesu aplikace, která se trasuje, a *%u* je jedinečné číslo pro rozlišení souborů mezi podprocesy spouštějícími trasování pod různými zavaděči tříd Java .

Není-li ID procesu k dispozici, vygeneruje se náhodné číslo s předponou s písmenem `f`. Chcete-li zahrnout ID procesu do zadaného názvu souboru, použijte řetězec `%PID%`.

Zadáte-li alternativní adresář, musí existovat a musíte mít pro tento adresář oprávnění k zápisu. Pokud nemáte oprávnění k zápisu, výstup trasování se zapíše do souboru `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 nebo IBM MQ classes for Jakarta Messaging.

Poznámka: Můžete zahrnout balík, ale pak vyloučit podbalíky tohoto balíku. Pokud například zahrnete balík `a.b` a vyloučíte balík `a.b.x`, trasování zahrne 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 vyloučí z trasování žádné balíky a třídy v IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging .

Poznámka: Můžete vyloučit balík, ale pak zahrnout podbalíky tohoto balíku. Pokud například vyloučíte balík `a.b` a zahrnete balík `a.b.x`, trasování zahrne 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 uvedena, na stejné úrovni, jak je zahrnutá, tak vyloučená.

com.ibm.msg.client.commonservices.trace.maxBytes = *maxArrayBajty*
`maxArrayBytes` je maximální počet bajtů, které jsou trasovány z libovolného bajtového pole.

Je-li parametr *`maxArrayBytes`* nastaven na kladné celé číslo, omezuje počet bajtů v bajtovém poli, které jsou zapsány do trasovacího souboru. Osekne bajtové pole po zapsání *`maxArrayBytes`* . Nastavení *`maxArrayBytes`* zmenšuje velikost výsledného trasovacího souboru a snižuje vliv trasování na výkon aplikace.

Hodnota 0 pro tuto vlastnost znamená, že do trasovacího souboru není odeslán žádný obsah žádného bajtového pole.

Výchozí hodnota je -1, která odebere jakýkoli limit počtu bajtů v bajtovém poli, které jsou odeslány do trasovacího souboru.

com.ibm.msg.client.commonservices.trace.limit = *maxTraceBajty*
maxTraceBytes je maximální počet bajtů zapsaných do výstupního souboru trasování.

maxTraceBytes pracuje s *traceCycles*. Pokud se počet zapsaných bajtů trasování blíží limitu, soubor se zavře a spustí se nový výstupní soubor trasování.

Hodnota 0 znamená, že výstupní soubor trasování má nulovou délku. Výchozí hodnota je -1, což znamená, že množství dat, která se mají zapsat do výstupního souboru trasování, je neomezené.

com.ibm.msg.client.commonservices.trace.count = *traceCycles*
traceCycles je počet výstupních souborů trasování, které se mají procházet.

Pokud aktuální výstupní soubor trasování dosáhne limitu uvedeného parametrem *maxTraceBytes*, soubor se zavře. Další trasovací výstup se zapíše do dalšího trasovacího výstupního souboru v pořadí. Každý výstupní soubor trasování je rozlišen číselnou příponou připojenou k názvu souboru. Aktuální nebo nejnovější výstupní soubor trasování je mqjms.trc.0, další nejnovější výstupní soubor trasování je mqjms.trc.1. Starší trasovací soubory následují stejný vzor číslování až do limitu.

Výchozí hodnota *traceCycles* je 1. Pokud je *traceCycles* 1, když aktuální výstupní soubor trasování dosáhne své maximální velikosti, soubor se zavře a odstraní. Spustí se 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*
traceParameters řídí, zda jsou parametry metody a návratové hodnoty zahrnuty do trasování.

traceParameters standardně zobrazuje TRUE. Je-li parametr *traceParameters* nastaven na hodnotu FALSE, budou trasovány pouze podpisy metod.

com.ibm.msg.client.commonservices.trace.startup = *spuštění*
Existuje fáze inicializace IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging , během které jsou prostředky přiděleny. Hlavní prostředek trasování je inicializován během fáze alokace prostředků.

Je-li parametr *startup* nastaven na hodnotu TRUE, použije se trasování spuštění. Informace o trasování jsou okamžitě vytvořeny a zahrnují nastavení všech komponent, včetně samotného prostředku trasování. Informace o trasování spuštění lze použít k diagnostice problémů s konfigurací. Informace o trasování spuštění se vždy zapisují do souboru System.err.

startup standardně zobrazuje FALSE.

Produkt *startup* je zkontrolován před dokončením inicializace. Z tohoto důvodu zadejte vlastnost na příkazovém řádku pouze jako systémovou vlastnost Java . Neuvádějte jej v konfiguračním souboru IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging .

com.ibm.msg.client.commonservices.trace.compress = *compressedTrace*
Nastavte *compressedTrace* na TRUE , chcete-li komprimovat výstup trasování.

Výchozí hodnota *compressedTrace* je FALSE.

Je-li parametr *compressedTrace* nastaven na hodnotu TRUE, výstup trasování je komprimován. Výchozí název výstupního souboru trasování má příponu .trz. Je-li komprese nastavena na hodnotu FALSE, což je výchozí hodnota, soubor má příponu .trc , která označuje, že je nekomprimovaný. Avšak pokud byl název souboru pro výstup trasování uveden v souboru *traceOutputName* , použije se tento název; pro soubor se nepoužije žádná přípona.

Komprimovaný výstup trasování je menší než nekomprimovaný. Protože je méně I/O, lze jej zapsat rychleji než nekomprimované trasování. Komprimované trasování má menší vliv na výkon IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging než nekomprimované trasování.

Jsou-li nastaveny hodnoty *maxTraceBytes* a *traceCycles* , vytvoří se místo více nestrukturovaných souborů více komprimovaných trasovacích souborů.

Pokud příkaz IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging skončí neřízeným způsobem, nemusí být komprimovaný trasovací soubor platný. Z tohoto důvodu musí být komprese trasování použita pouze v případě, že se produkt IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging ukončí řízeným způsobem. Kompresi trasování používejte pouze v případě, že vyšetřované problémy nezpůsobí neočekávané zastavení prostředí JVM. Nepoužívejte kompresi trasování při diagnostice problémů, které mohou vést k `System.Halt()` vypnutí nebo nekontrolovanému nekontrolovanému ukončení prostředí JVM.

com.ibm.msg.client.commonservices.trace.level = *traceLevel*
traceLevel uvádí úroveň filtrování pro trasování. Definované úrovně trasování jsou 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, bude do trasování zapsán libovolný bod trasování s definovanou úrovní TRACE_EXCEPTION, TRACE_WARNINGnebo TRACE_INFO . Všechny ostatní trasovací body jsou vyloučeny.

com.ibm.msg.client.commonservices.trace.standalone = *standaloneTrace*

standaloneTrace řídí, zda se služba trasování klienta IBM MQ JMS používá v prostředí WebSphere Application Server .

Je-li parametr *standaloneTrace* nastaven na hodnotu TRUE, vlastnosti trasování klienta IBM MQ JMS se používají k určení konfigurace trasování.

Je-li parametr *standaloneTrace* nastaven na hodnotu FALSEa klient IBM MQ JMS je spuštěn v kontejneru WebSphere Application Server , použije se trasovací služba WebSphere Application Server . Generované informace o trasování závisí na nastavení trasování aplikačního serveru.

Výchozí hodnota *standaloneTrace* je FALSE.

Oddíl protokolování
Použijte sekci Protokolování ke konfiguraci prostředku protokolu IBM MQ classes for JMS .

V sekci Protokolování lze zahrnout následující vlastnosti:

com.ibm.msg.client.commonservices.log.outputName = cesta
Název souboru protokolu, který používá prostředek protokolu IBM MQ classes for JMS . Výchozí hodnota je `mqjms.log`, která se zapíše do aktuálního pracovního adresáře pro běhové prostředí Java Runtime Environment, ve kterém běží produkt IBM MQ classes for JMS .

Vlastnost může mít jednu z následujících hodnot:

- jeden název cesty
- seznam názvů cest oddělených čárkami (všechna data jsou protokolována do všech souborů)

Každý název cesty může být absolutní nebo relativní název cesty nebo:

"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 jsou protokolovány z jakéhokoli volání do dat zprávy protokolu.

Kladné celé číslo
Data se zapisují do této hodnoty bajtů na volání protokolu.

0
Nejsou zapsána žádná data.

-1
Zapisují se neomezená data (výchozí).

com.ibm.msg.client.commonservices.log.limit
Maximální počet bajtů zapsaných do libovolného 1 souboru protokolu (předvolba je 262144).

Kladné celé číslo
Data jsou zapsána až do této hodnoty bajtů na soubor protokolu.

0
Nejsou zapsána žádná data.

-1
Zapisují se neomezená data.

com.ibm.msg.client.commonservices.log.count
Počet souborů protokolu, které se mají procházet. Jakmile každý soubor dosáhne trasování `com.ibm.msg.client.commonservices.trace.limit` , začne v dalším souboru, předvolba je 3.

Kladné celé číslo
Počet souborů, které se mají procházet.

0
Jeden soubor.

Java Specifika SE-sekce
Použijte sekci Java Specifika SE 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 po vygenerování souboru FDC produktem IBM MQ classes for JMS . Je-li tato hodnota nastavena na TRUE, je soubor jádra Javavytvořen v pracovním adresáři běhového prostředí Java , ve kterém je spuštěn produkt IBM MQ classes for JMS .

TRUE
Vygenerujte jádro Javav závislosti na schopnosti běhového prostředí Java .

FALSE
Negenerovat jádro Java; toto je výchozí hodnota.

IBM MQ Vlastnosti sekce
Pomocí sekce IBM MQ Properties můžete nastavit vlastnosti, které ovlivňují způsob IBM MQ classes for JMS interakce s IBM MQ.

com.ibm.msg.client.wmq.compat.base.internal.MQQueue.smallMsgsBufferReductionThreshold
Když se aplikace, která používá produkt IBM MQ classes for JMS , připojuje ke správci front IBM MQ pomocí režimu migrace poskytovatele systému zpráv IBM MQ , použije produkt IBM MQ classes for JMS při přijímání zpráv výchozí velikost vyrovnávací paměti 4 kB. Pokud je zpráva, kterou se aplikace pokouší získat, větší než 4 kB, produkt IBM MQ classes for JMS změní velikost vyrovnávací paměti tak, aby byla dostatečně velká pro uložení zprávy. Větší velikost vyrovnávací paměti se pak použije při přijetí následných zpráv.

Tato vlastnost určuje, kdy je velikost vyrovnávací paměti snížena zpět na 4 kB. Standardně, když je přijato deset po sobě jdoucích zpráv, které jsou menší než větší velikost vyrovnávací paměti, velikost

vyrovnávací paměti se sníží zpět na 4 kB. Chcete-li resetovat 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, která používá produkt IBM MQ classes for JMS , připojuje ke správci front IBM MQ pomocí normálního režimu poskytovatele systému zpráv IBM MQ , lze nastavit vlastnost `receiveConversionCCSID` tak, aby potlačovala výchozí hodnotu CCSID ve struktuře MQMD, která se používá pro příjem zpráv ze správce front. Standardně modul MQMD obsahuje pole CCSID nastavené na hodnotu 1208, ale tuto hodnotu lze změnit například v případě, že 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.

Sekce specifik režimu klienta

Pomocí specifické sekce režimu klienta můžete určit vlastnosti, které se použijí při připojení produktu IBM MQ classes for JMS ke správci front, který používá přenos CLIENT.

com.ibm.mq.polling.RemoteRequestEntry

Uvádí interval výzev, který produkt 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 s nižšími hodnotami se zachází stejně jako s touto minimální hodnotou.

Vlastnosti použité ke konfiguraci chování klienta JMS

Pomocí těchto vlastností můžete konfigurovat chování klienta JMS .

com.ibm.mq.jms.SupportMQExtensions TRUE|FALSE

Specifikace JMS 2.0 zavádí změny způsobu práce s určitým chováním. IBM MQ 8.0 zahrnuje vlastnost `com.ibm.mq.jms.SupportMQExtensions`, kterou lze nastavit na hodnotu *TRUE*, aby se tyto změněné chování vrátily zpět na předchozí implementace. Opětovné vrácení změněného chování může být nezbytné pro aplikace JMS 2.0 a také pro některé aplikace, 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 vráceny zpět nastavením volby SupportMQExtensions na hodnotu TRUE:

Priorita zprávy

Zprávám lze přiřadit prioritu *0* - *9*. Před produktem JMS 2.0mohou zprávy také používat hodnotu *-1*, která označuje, že je použita výchozí priorita fronty. JMS 2.0 nepovoluje nastavení priority zprávy *-1* . Zapnutím `SupportMQExtensions` umožníte použití hodnoty *-1* .

ID klienta

Specifikace JMS 2.0 vyžaduje, aby byla při vytváření připojení kontrolována jedinečnost ID klientů s jinou hodnotou než Null. Zapnutí `SupportMQExtensions`znamená, že tento požadavek nebude zohledněn a že ID klienta lze znovu použít.

`NoLocal`

Specifikace JMS 2.0 vyžaduje, aby při zapnutí této konstanty spotřebitel nemohl přijímat zprávy publikované stejným ID klienta. Před JMS 2.0byl tento atribut nastaven na odběrateli,

aby zabránil přijímání zpráv, které jsou publikovány jeho vlastním připojením. Zapnutí produktu `SupportMQExtensions` vrátí toto chování zpět na 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 typu Bajty nebo Proud nastavit stav zprávy, která byla právě odeslána, buď jen pro 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 pro zápis pouze po odeslání. Toto je výchozí hodnota.

Související pojmy

“Vlastnost SupportMQExtensions” na stránce 317
Specifikace JMS 2.0 zavedla změny způsobu práce s určitým chováním. Produkt IBM MQ 8.0 a novější obsahuje vlastnost **com.ibm.mq.jms.SupportMQExtensions**, kterou lze nastavit na hodnotu TRUE , aby se tyto změněné chování vrátily zpět na předchozí implementace.

Konfigurace STEPLIB pro IBM MQ classes for JMS on z/OS
V systému z/OSmusí knihovna STEPLIB použitá za běhu obsahovat knihovny IBM MQ SCSQAUTH a SCSQANLE. Tyto knihovny zadejte ve spouštěcím JCL nebo pomocí souboru `.profile` .

From z/OS UNIX System Services, you can add these using a line in your `.profile` as shown in the following code snippet, replacing `thlqual` with the high-level data set qualifier that you chose when installing IBM MQ:

```
export STEPLIB=thlqual.SCSQAUTH:thlqual.SCSQANLE:$STEPLIB
```

V jiných prostředích obvykle potřebujete upravit spouštěcí JCL tak, aby zahrnoval SCSQAUTH a SCSQANLE ve 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 nástroji IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging.

Mnoho velkých vývojových organizací používá tyto nástroje k centrální správě úložišť knihoven třetích stran.

Soubory IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging se skládají z několika souborů JAR. Při vývoji jazykových aplikací Java pomocí tohoto rozhraní API je na počítači, na kterém je aplikace vyvíjena, vyžadována instalace produktu IBM MQ Server, IBM MQ Client nebo IBM MQ Client SupportPac .

Chcete-li použít takový nástroj a přidat soubory JAR, které tvoří soubor IBM MQ classes for JMS , do centrálně spravovaného úložiště, musíte dodržet následující body:

- Úložiště nebo kontejner musí být k dispozici pouze vývojářům ve vaší organizaci. Distribuce mimo organizaci není povolena.
- Úložiště musí obsahovat úplnou a konzistentní sadu souborů JAR z jednoho vydání IBM MQ nebo opravné sady.
- Jste zodpovědní za aktualizaci úložiště s jakoukoli údržbou poskytovanou podporou IBM .

Do úložiště je třeba nainstalovat následující soubory JAR:

- JMS 2.0 `com.ibm.mq.allclient.jar` a `jms.jar` jsou povinné, pokud používáte IBM MQ classes for JMS.
- JM 3.0 `com.ibm.mq.jakarta.client.jar` a `jakarta.jms-api.jar` jsou povinné, pokud používáte IBM MQ classes for Jakarta Messaging.
- Parametr `fscontext.jar` je povinný, pokud používáte produkt IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging a přistupujete k objektům spravovaným produktem JMS , které jsou uloženy v kontextu rozhraní JNDI systému souborů.
- `providerutil.jar` je povinný, pokud používáte IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging a přistupujete k spravovaným objektům JMS , které jsou uloženy v kontextu JNDI systému souborů.
- Poskytovatel zabezpečení Bouncy Castle a soubory JAR podpory CMS jsou nezbytné pro podporu jiných prostředí JRE nežIBM . Další informace naleznete v tématu Podpora jiných prostředí JRE nežIBM.

Spuštění aplikací IBM MQ classes for JMS pod Java security manager

IBM MQ classes for JMS lze spustit s povoleným Java security manager . Chcete-li úspěšně spustit aplikace s povoleným produktem Java security manager , musíte nakonfigurovat prostředí Java Virtual Machine (JVM) pomocí vhodného konfiguračního souboru zásad.

Nejjednodušším způsobem, jak vytvořit vhodný soubor definice zásady, je změnit konfigurační soubor zásady dodaný s vaším prostředím Java runtime environment (JRE). Na většině systémů je tento soubor v adresáři `lib/security/java.policy` relativně k vašemu adresáři JRE. Konfigurační soubor zásad můžete upravit buď pomocí upřednostňovaného editoru, nebo pomocí programu nástroje zásad dodaného s prostředím JRE.

Příklad konfiguračního souboru zásad

Zde je uveden příklad konfiguračního souboru zásad, který umožňuje úspěšné spuštění produktu IBM MQ classes for JMS pod výchozím správcem zabezpečení. Tento soubor bude třeba upravit, aby bylo možné určit umístění určitých souborů a adresářů: *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ , *MQ_DATA_DIRECTORY* představuje umístění datového adresáře produktu MQ a *QM_NAME* je název správce front, pro kterého je konfigurován přístup.

```
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 příkladu příkaz `grant` obsahuje oprávnění vyžadovaná produktem IBM MQ classes for JMS. Chcete-li použít tyto příkazy udělení oprávnění v konfiguračním souboru zásad, možná budete muset upravit názvy cest v závislosti na tom, kam jste nainstalovali produkt IBM MQ classes for JMS a kam ukládáte aplikace.

Ukázkové aplikace dodávané s produktem IBM MQ classes for JMSa skripty pro jejich spuštění nepovolují správce zabezpečení.

Důležité:

Prostředek trasování IBM MQ classes for JMS vyžaduje další oprávnění, protože provádí další dotazy na systémové vlastnosti a také další operace systému souborů.

Vhodný soubor zásad zabezpečení šablony pro spuštění ve správci zabezpečení s povoleným trasováním je uveden v adresáři `samples/wmqjava` instalace produktu IBM MQ jako `example.security.policy`.

Nastavení po instalaci pro aplikace IBM MQ classes for JMS

V tomto tématu jsou uvedeny informace o potřebách aplikací IBM MQ classes for JMS s oprávněními pro přístup k prostředkům správce front. Zavádí také režimy připojení a popisuje, jak nakonfigurovat správce front tak, aby se aplikace mohly připojovat v režimu klienta.

Nezapomeňte zkontrolovat soubor IBM MQ readme. Může obsahovat informace, které nahrazují informace v tomto tématu.

Objekty používané produktem JMS , které vyžadují autorizaci pro neprivilegované uživatele

Neprivilegovaní uživatelé potřebují udělit oprávnění 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 MQnaleznete v tématu Nastavení zabezpečení.

Aplikace IBM MQ classes for JMS potřebují oprávnění `connect` a `inq` ke správci front. Pomocí řídicího příkazu **setmqaut** můžete nastavit příslušné autorizace, například:

```
setmqaut -m QM1 -t qmgr -g jmsappsgroup +connect +inq
```

Pro doménu typu point-to-point jsou požadována následující oprávnění:

- Fronty používané objekty MessageProducer potřebují oprávnění `put` .
- Fronty používané objekty MessageConsumer a QueueBrowser potřebují oprávnění `get`, `inq`a `browse` .
- Metoda QueueSession.createTemporaryQueue () potřebuje přístup k modelové frontě určené vlastností TEMPMODEL objektu továrny QueueConnection. Standardně je tato modelová fronta SYSTEM.TEMP.MODEL.QUEUE.

Pokud jsou některé z těchto front alias fronty, jejich cílové fronty vyžadují oprávnění k dotazování. Je-li cílovou frontou fronta klastru, vyžaduje také oprávnění k procházení.

Pro doménu publikování/odběru se používají následující fronty, pokud se IBM MQ classes for JMS připojují ke správci front IBM MQ v režimu migrace poskytovatele systému zpráv IBM MQ :

- SYSTEM.JMS.ADMIN.QUEUE
- SYSTEM.JMS.REPORT.QUEUE
- SYSTEM.JMS.MODEL.QUEUE
- SYSTEM.JMS.PS.STATUS.QUEUE
- SYSTEM.JMS.ND.SUBSCRIBER.QUEUE
- SYSTEM.JMS.D.SUBSCRIBER.QUEUE
- SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE
- SYSTEM.JMS.D.CC.SUBSCRIBER.QUEUE
- SYSTEM.BROKER.CONTROL.QUEUE

Další informace o režimu migrace poskytovatele systému zpráv IBM MQ naleznete v tématu Konfigurace vlastnosti JMS **PROVIDERVERSION**

Pokud se navíc produkt IBM MQ classes for JMS připojuje ke správci front v tomto režimu, každá aplikace, která publikuje zprávy, potřebuje přístup k frontě proudu určené továrnou nebo objektem tématu TopicConnection. Standardně je tato fronta SYSTEM.BROKER.DEFAULT.STREAM.

Pokud používáte ConnectionConsumer, IBM MQ Resource Adapter nebo poskytovatele systému zpráv WebSphere Application Server IBM MQ , může být zapotřebí další autorizace.

Fronty, které mají být čteny konzumentem připojení ConnectionConsumer , musí mít oprávnění `get`, `inq`a `browse` . Fronta nedoručených zpráv systému a jakákoli fronta nedoručených zpráv nebo fronta sestav používaná konzumentem připojení ConnectionConsumer musí mít oprávnění `put` a `passall` .

Pokud aplikace používá k provádění systému zpráv publikování/odběru poskytovatele systému zpráv IBM MQ v normálním režimu, využívá funkce integrovaného publikování/odběru poskytované správcem front. Informace o zabezpečení témat a front, které se používají, naleznete v tématu Zabezpečení publikování/odběru .

Režimy připojení pro IBM MQ classes for JMS
Aplikace IBM MQ classes for JMS se může připojit ke správci front v režimu klienta nebo v režimu vazeb. V režimu klienta se produkt IBM MQ classes for JMS připojuje ke správci front prostřednictvím protokolu TCP/IP. V režimu vazeb se produkt IBM MQ classes for JMS připojuje přímo ke správci front pomocí rozhraní JNI (Java Native Interface).

Aplikace spuštěná v systému WebSphere Application Server v systému z/OS se může připojit ke správci front v režimu vazeb nebo v režimu klienta, ale aplikace spuštěná v jakémkoli jiném prostředí v systému z/OS se může připojit ke správci front pouze v režimu vazeb. Aplikace spuštěná na libovolné jiné platformě se může připojit ke správci front v režimu vazeb nebo v režimu klienta.

Můžete použít aktuální nebo 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 kombinujete různé verze, funkce je omezena na úroveň předchozí verze.

Následující sekce podrobněji popisují jednotlivé režimy připojení.

Režim klienta

Chcete-li se připojit ke správci front v režimu klienta, může být aplikace IBM MQ classes for JMS spuštěna ve stejném systému, ve kterém je spuštěn správce front, nebo v 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 IBM MQ classes for JMS spuštěna ve stejném systému, ve kterém 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, musí být IBM MQ classes for JMS spuštěn v prostředí, které má přístup ke knihovnám nativního rozhraní produktu IBM MQ Java ; další informace viz “Konfigurace knihoven JNI (Java Native Interface)” na stránce 92 .

Produkt IBM MQ classes for JMS podporuje následující hodnoty pro volbu *ConnectOption*:

- MQCNO_FASTPATH_BINDING
- MQCNO_STANDARDNÍ_VAZBA
- MQCNO_SHARED_BINDING
- MQCNO_ISOLATED_BINDING
- MQCNO_RESTRICT_CONN_TAG_QSG
- MQCNO_RESTRICT_CONN_TAG_Q_MGR

Chcete-li změnit volby připojení používané produktem IBM MQ classes for JMS, upravte vlastnost Továrna připojení CONNOPT.

Další informace o volbách připojení viz “Připojení ke správci front pomocí volání MQCONNX” na stránce 715

Chcete-li použít přenos vazeb, musí používané běhové prostředí Java podporovat identifikátor kódované znakové sady (CCSID) správce front, ke kterému se produkt IBM MQ classes for JMS připojuje.

Podrobnosti o tom, jak zjistit, jaké CCSID podporuje běhové prostředí Java , naleznete v produktu IBM MQ FDC s ID sondy 21 generovaném při použití tříd IBM MQ V7 pro třídy Java nebo IBM MQ V7 pro produkt JMS.

Konfigurace správce front tak, aby se aplikace IBM MQ classes for JMS mohly připojovat v režimu klienta
Chcete-li nakonfigurovat správce front tak, aby se aplikace IBM MQ classes for JMS mohly připojovat v režimu klienta, musíte vytvořit definici kanálu připojení serveru a spustit modul listener.

Vytvoření definice kanálu připojení serveru

Na všech platformách můžete k vytvoření definice kanálu připojení serveru použít příkaz MQSC DEFINE CHANNEL. 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 také vytvořit definici kanálu připojení serveru pomocí IBM MQ Explorer.

z/OS V systému z/OS můžete pomocí operací a ovládacích panelů vytvořit definici kanálu připojení serveru.

Název kanálu (JAVA.CHANNEL v předchozích příkladech) musí být stejný jako název kanálu určený vlastností CHANNEL továrny připojení, kterou vaše aplikace používá pro připojení ke správci front. Výchozí hodnota vlastnosti CHANNEL je SYSTEM.DEF.SVRCONN.

Spuštění modulu listener

Pokud ještě není spuštěn modul listener pro vašeho správce front, musíte ho spustit.

Multi V systému Multiplatformsmůžete použít příkaz MQSC START LISTENER ke spuštění modulu listener po prvním vytvoření objektu modulu listener pomocí příkazu MQSC DEFINE LISTENER, jak ukazuje následující příklad:

```
DEFINE LISTENER(LISTENER.TCP) TRPTYPE(TCP) PORT(1414)
START LISTENER(LISTENER.TCP)
```

z/OS V systému z/OSpoužijete pouze příkaz START LISTENER, jako v následujícím příkladu, ale uvědomte si, že před spuštěním modulu listener musí být spuštěn adresní prostor inicializátoru kanálu:

```
START LISTENER TRPTYPE(TCP) PORT(1414)
```

IBM i V systému IBM můžete ke spuštění modulu listener použít také příkaz CL STRMQMLSR, jako v následujícím příkladu:

```
STRMQMLSR PORT(1414) MQMNAME(QMGRNAME)
```

V tomto příkazu je *QMGRNAME* název vašeho správce front.

ALW V systému AIX, Linux, and Windowsmůžete také použít řídicí příkaz **runmqlsr** ke spuštění modulu listener, 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 IBM MQ Explorer.

z/OS V systému z/OSmůžete ke spuštění modulu listener také použít operace a ovládací panely.

Číslo portu, na kterém naslouchá modul listener, musí být stejné jako číslo portu určené vlastností PORT továrny připojení, kterou vaše aplikace používá pro připojení ke správci front. Výchozí hodnota vlastnosti PORT je 1414.

Dvoubodový IVT pro IBM MQ classes for JMS

Program IVT (point-to-point installation verification test) je dodáván s produktem IBM MQ classes for JMS a produktem IBM MQ classes for Jakarta Messaging. Program se připojí ke správci front v režimu vazeb nebo v režimu klienta a odešle zprávu do fronty s názvem SYSTEM.DEFAULT.LOCAL.QUEUEa poté obdrží 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.

Nejprve spusťte ověřovací test instalace bez použití rozhraní JNDI, protože test je samostatný a nevyžaduje použití adresářové služby. Popis spravovaných objektů naleznete v tématu Konfigurace objektů JMS pomocí nástroje pro administraci.

Ověřovací test dvoubodové instalace bez použití rozhraní JNDI

V tomto testu program IVT vytváří a konfiguruje všechny objekty, které vyžaduje, dynamicky za běhu a nepoužívá rozhraní JNDI.

Multi Na multiplatformách je poskytnut skript pro spuštění programu IVT. Skript se nazývá **IVTRun** na systémech AIX and Linux a **IVTRun.bat** na systému Windows. skript je umístěn v podadresáři `bin` instalačního adresáře IBM MQ classes for JMS . Cesta ke třídě musí obsahovat `com.ibm.mqjms.jar`.

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 podle popisu v části “Konfigurace správce front pro přijetí připojení klienta na platformě Multiplatforms” na stránce 1034. Všimněte si, že kanál, který se má použít, má výchozí hodnotu **SYSTEM.DEF.SVRCONN** a fronta, která se má použít, je **SYSTEM.DEFAULT.LOCAL.QUEUE**, pak zadejte následující příkaz:

```
IVTRun -nojndi -client -m qmgr -host hostname [-port port ] [-channel channel ]
[-v providerVersion ] [-ccsid ccsid ] [-t]
```

z/OS Na systémech z/OS není k dispozici žádný ekvivalentní skript. Namísto toho spustíte IVT v režimu vazeb přímým vyvoláním třídy Java . V systému z/OSsi můžete vybrat mezi dvěma funkčně identickými instancemi programu IVT:

- `com.ibm.mq.jms.MQJMSIVT`, který je k dispozici s IBM MQ classes for JMS (JMS 2.0). Chcete-li použít tento program, cesta ke třídě musí obsahovat `com.ibm.mqjms.jar` nebo `com.ibm.mq.allclient.jar`.
- `com.ibm.mq.jakarta.jms.MQJMSIVT`, který je k dispozici s IBM MQ classes for Jakarta Messaging (Jakarta Messaging 3.0). Chcete-li použít tento program, cesta ke třídě musí obsahovat `com.ibm.mq.jakarta.client.jar`.

Chcete-li spustit test v režimu vazeb v systému z/OS, zadejte následující příkaz:

```
java com.ibm.mq.jms.MQJMSIVT -nojndi [-m qmgr ] [-v providerVersion ] [-t]
```

Parametry v příkazech mají následující význam:

-m *správce front*
Název správce front, ke kterému se program IVT připojuje. Spustíte-li test v režimu vazeb a vynecháte-li tento parametr, program IVT se připojí k výchozímu správci front.

-host *hostname*
Název hostitele nebo adresa IP systému, na kterém je spuštěn správce front.

-port *port*
Číslo portu, na kterém naslouchá modul listener správce front. Výchozí hodnota je 1414.

-channel ***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 má program IVT připojit.

Tento parametr se používá k nastavení vlastnosti PROVIDERVERSION objektu továrny MQQueueConnectiona má stejné platné hodnoty jako vlastnost PROVIDERVERSION. Další informace o tomto parametru, včetně jeho platných hodnot, viz JMS: změny vlastnosti PROVIDERVERSION a popis vlastnosti PROVIDERVERSION v Properties IBM MQ classes for JMS objektů.

Výchozí hodnota je `unspecified`.

-ccsid ***ccsid***
Identifikátor (CCSID) kódované znakové sady nebo kódové stránky, která má být použita připojením. Výchozí hodnota je 819.

-t
Trasování je povoleno. Standardně je trasování zakázáno.

Úspěšný test vytvoří výstup podobný následujícímu ukázkovému výstupu:

```
5724-H72, 5655-R36, 5724-L26, 5655-L82 (c) Copyright IBM Corp. 2008, 2024. All
Rights Reserved.
WebSphere MQ classes for Java(tm) Message Service 7.0
Installation Verification Test

Creating a QueueConnectionFactory
Creating a Connection
Creating a Session
Creating a Queue
Creating a QueueSender
Creating a QueueReceiver
Creating a TextMessage
Sending the message to SYSTEM.DEFAULT.LOCAL.QUEUE
Reading the message back again

Got message
JMSMessage class: jms_text
JMSType:     null
JMSDeliveryMode: 2
JMSExpiration:  0
JMSPriority:   4
JMSMessageID:   ID:414d5120514d5f6d627720202020202001edb14620005e03
JMSTimestamp:   1187170264000
JMSCorrelationID: null
JMSDestination:  queue:///SYSTEM.DEFAULT.LOCAL.QUEUE
JMSReplyTo:    null
JMSRedelivered:  false
JMSXUserID: mwhite
JMS_IBM_Encoding: 273
JMS_IBM_PutApplType: 28
JMSXAppID: 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
```

Ověřovací test dvoubodové instalace pomocí rozhraní JNDI

Multi

V tomto testu používá program IVT k načtení spravovaných objektů z adresářové služby rozhraní JNDI.

Před spuštěním testu je třeba konfigurovat adresářovou službu založenou na serveru LDAP (Lightweight Directory Access Protocol) nebo lokálním systému souborů. Musíte také nakonfigurovat nástroj pro administraci produktu IBM MQ JMS tak, aby mohl používat adresářovou službu k ukládání spravovaných objektů. Další informace o těchto předpokladech viz “Předpoklady pro IBM MQ classes for JMS” na stránce 84. Informace o konfiguraci nástroje pro administraci produktu IBM MQ JMS naleznete v tématu Konfigurace JMS nástroje pro administraci.

Program IVT musí být schopen použít rozhraní JNDI k načtení objektu továrny MQQueueConnectiona objektu MQQueue z adresářové služby. Pro vytvoření těchto spravovaných objektů je k dispozici skript. Skript se nazývá IVTSetup na systémech AIX and Linux a IVTSetup.bat na systému Windowsa je v podadresáři `bin` instalačního adresáře IBM MQ classes for JMS . Chcete-li spustit skript, zadejte následující příkaz:

```
IVTSetup
```

Skript vyvolá nástroj pro administraci produktu IBM MQ JMS a vytvoří spravované objekty.

Objekt továrny MQQueueConnectionje svázán s názvem ivtQCF a je vytvořen s výchozími hodnotami pro všechny jeho vlastnosti, což znamená, že program IVT je spuštěn v režimu vazeb a připojuje se k výchozímu správci front. Chcete-li spustit program IVT v režimu klienta nebo se připojit ke správci front, který není výchozím správcem front, je třeba změnit příslušné vlastnosti objektu továrny MQQueueConnectionpomocí nástroje pro administraci produktu IBM MQ JMS nebo IBM MQ Explorer . Informace o použití administračního nástroje produktu IBM MQ Explorer JMS naleznete v tématu Konfigurace objektů JMS pomocí administračního nástroje. Chcete-li získat informace o tom, jak používat produkt IBM MQ Explorer, prohlédněte si téma Úvod do produktu IBM MQ Explorer nebo nápovědu poskytnutou s produktem IBM MQ Explorer.

Objekt MQQueue je svázán s názvem ivtQ a je vytvořen s výchozími hodnotami pro všechny jeho vlastnosti s výjimkou vlastnosti QUEUE, která má hodnotu SYSTEM.DEFAULT.LOCAL.QUEUE.

Po vytvoření spravovaných objektů 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*"
Lokátor jednotného prostředku (URL) adresářové služby. 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

Nezapomeňte, že URL musíte uzavřít do uvozovek (").

-icf *initCtxFact*
Název třídy továrny počátečního kontextu, která musí být jednou 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. Standardně je trasování zakázáno.

Úspěšný test vytvoří podobný výstup 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.

Ačkoli to není nezbytně nutné, doporučuje se po testu uklidit odstraněním spravovaných objektů vytvořených skriptem IVTSetup. K tomuto účelu je poskytnut skript. Skript se nazývá IVTTidy na systémech AIX and Linux a IVTTidy.bat na systému Windowsa je v podadresáři `bin` instalačního adresáře IBM MQ classes for JMS .

Určení problému pro ověřovací test dvoubodové instalace

Ověřovací test instalace může selhat z následujících příčin:

- Pokud program IVT zapíše zprávu označující, že nemůže najít třídu, zkontrolujte, zda je správně nastavena vaše cesta ke třídám, jak je popsáno v tématu “Nastavení proměnných prostředí pro IBM MQ classes for JMS/Jakarta Messaging” na stránce 90.
- Test může selhat s následující zprávou:

  ```
  Failed to connect to queue manager ' qmgr ' with connection mode ' connMode '
  and host name ' hostname '
  ```

  a přidružený kód příčiny 2059. Proměnné ve zprávě mají následující význam:

  QMGR
  Název správce front, ke kterému se program IVT pokouší připojit. Tato vložení zprávy je prázdné, 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, na kterém 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, zkontrolujte, zda je tento správce front 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. Pokud fronta existuje, program IVT ji nemůže otevřít, 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 server LDAP není správně nakonfigurován. Buď není server LDAP konfigurován pro ukládání objektů Java , nebo nejsou správná oprávnění k objektům nebo příponě. Další nápovědu k této situaci naleznete v dokumentaci k 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 z vašeho systému. Podrobnosti viz “Konfigurace správce front pro přijetí připojení klienta na platformě Multiplatforms” na stránce 1034.

IVT publikování/odběru 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 režimu vazby, nebo v režimu klienta, přihlásí se k odběru tématu, publikuje zprávu v tématu a poté obdrží 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.

Nejprve spusťte ověřovací test instalace bez použití rozhraní JNDI, protože test je samostatný a nevyžaduje použití adresářové služby. Popis spravovaných objektů naleznete v tématu Konfigurace objektů JMS pomocí nástroje pro administraci.

Ověřovací test instalace publikování/odběru bez použití rozhraní JNDI

V tomto testu program IVT vytváří a konfiguruje všechny objekty, které vyžaduje, dynamicky za běhu a nepoužívá rozhraní JNDI.

Je poskytnut skript pro spuštění programu IVT. Skript se nazývá PSIVTRun na systémech AIX and Linux a PSIVTRun.bat na systému Windowsa je v podadresáři `bin` instalačního adresáře 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 podle popisu v tématu “Konfigurace správce front pro přijetí připojení klienta na platformě Multiplatforms” na stránce 1034 a poznamenejte si, že kanál, který má být použit, má výchozí hodnotu SYSTEM.DEF.SVRCONN, zadejte následující 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. Spustíte-li test v režimu vazeb a vynecháte-li tento parametr, program IVT se připojí k výchozímu správci front.

-host *hostname*
Název hostitele nebo adresa IP systému, na kterém je spuštěn správce front.

-port *port*
Číslo portu, na kterém naslouchá modul listener správce front. Výchozí hodnota je 1414.

-channel *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 verzi správce front s číslem v 7 nebo vyšším.

-v *providerVersion*
Úroveň vydání správce front, ke kterému se má program IVT připojit.

Tento parametr se používá k nastavení vlastnosti PROVIDERVERSION objektu továrny MQTopicConnectiona má stejné platné hodnoty jako vlastnost PROVIDERVERSION. Další informace

o tomto parametru, včetně jeho platných hodnot, naleznete v popisu vlastnosti PROVIDERVERSION v části Vlastnosti IBM MQ classes for JMS objektů.

Výchozí hodnota je `unspecified`.

-ccsid ***ccsid***
Identifikátor (CCSID) kódované znakové sady nebo kódové stránky, která má být použita připojením. Výchozí hodnota je 819.

-t
Trasování je povoleno. Standardně je trasování zakázáno.

Úspěšný test vytvoří výstup podobný následujícímu ukázkovému výstupu:

```
5724-H72, 5655-R36, 5724-L26, 5655-L82 (c) Copyright IBM Corp. 2008, 2024. All
Rights Reserved.
IBM MQ classes for Java 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:414d5120514d5f6d62772020202020201edb14620006706
JMSTimestamp:   1187182520203
JMSCorrelationID: ID:414d5120514d5f6d62772020202020201edb14620006704
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: 414D5143514D5F6D62772020202020201EDB14620006601
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 pomocí rozhraní JNDI

V tomto testu používá program IVT k načtení spravovaných objektů z adresářové služby rozhraní JNDI.

Před spuštěním testu je třeba konfigurovat adresářovou službu založenou na serveru LDAP (Lightweight Directory Access Protocol) nebo lokálním systému souborů. Musíte také nakonfigurovat nástroj pro administraci produktu IBM MQ JMS tak, aby mohl používat adresářovou službu k ukládání spravovaných objektů. Další informace o těchto předpokladech viz “Předpoklady pro IBM MQ classes for JMS” na

stránce 84. Informace o konfiguraci nástroje pro administraci produktu IBM MQ JMS naleznete v tématu Konfigurace JMS nástroje pro administraci.

Program IVT musí být schopen použít rozhraní JNDI k načtení objektu továrny MQTopicConnectiona objektu MQTopic z adresářové služby. Pro vytvoření těchto spravovaných objektů je k dispozici skript. Skript se nazývá IVTSetup na systémech AIX and Linux a IVTSetup.bat na systému Windowsa je v podadresáři `bin` instalačního adresáře IBM MQ classes for JMS . Chcete-li spustit skript, zadejte následující příkaz:

```
IVTSetup
```

Skript vyvolá nástroj pro administraci produktu IBM MQ JMS a vytvoří spravované objekty.

Objekt továrny MQTopicConnectionje svázán s názvem ivtTCF a je vytvořen s výchozími hodnotami pro všechny jeho vlastnosti, což znamená, že program IVT je spuštěn v režimu vazeb, připojuje se k výchozímu správci front a používá vestavěnou funkci publikování/odběru. Chcete-li, aby byl program IVT spouštěn v režimu klienta, připojte se ke správci front, který není výchozím správcem front, nebo použijte funkci IBM Integration Bus namísto vestavěné funkce publikování/odběru, musíte změnit příslušné vlastnosti objektu továrny MQTopicConnectionpomocí administračního nástroje produktu IBM MQ JMS nebo pomocí Průzkumníku IBM MQ . Informace o použití administračního nástroje produktu IBM MQ JMS naleznete v tématu Konfigurace objektů JMS pomocí administračního nástroje. Informace o použití Průzkumníku IBM MQ naleznete v nápovědě k Průzkumníku IBM MQ .

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.

Po vytvoření spravovaných objektů 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*"
Lokátor jednotného prostředku (URL) adresářové služby. 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

Nezapomeňte, že URL musíte uzavřít do uvozovek (").

-icf *initCtxFact*
Název třídy továrny počátečního kontextu, která musí být jednou 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. Standardně je trasování zakázáno.

Úspěšný test vytvoří podobný výstup 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.

Ačkoli to není nezbytně nutné, doporučuje se po testu uklidit odstraněním spravovaných objektů vytvořených skriptem IVTSetup. K tomuto účelu je poskytnut skript. Skript se nazývá IVTTidy na systémech AIX and Linux a IVTTidy.bat na systému Windowsa je v podadresáři `bin` instalačního adresáře IBM MQ classes for JMS .

Určení problému pro ověřovací test instalace publikování/odběru

Ověřovací test instalace může selhat z následujících příčin:

- Pokud program IVT zapíše zprávu označující, že nemůže najít třídu, zkontrolujte, zda je správně nastavena vaše cesta ke třídám, jak je popsáno v tématu "Nastavení proměnných prostředí pro IBM MQ classes for JMS/Jakarta Messaging" na stránce 90.
- Test může selhat s následující zprávou:

```
Failed to connect to queue manager ' qmgr ' with
connection mode ' connMode ' and host name ' hostname '
```

  a přidružený kód příčiny 2059. Proměnné ve zprávě mají následující význam:

  QMGR
  Název správce front, ke kterému se program IVT pokouší připojit. Tato vložení zprávy je prázdné, 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, na kterém 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, zkontrolujte, zda je tento správce front 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 server LDAP není správně nakonfigurován. Buď není server LDAP konfigurován pro ukládání objektů Java , nebo nejsou správná oprávnění k objektům nebo příponě. Další nápovědu k této situaci naleznete v dokumentaci k 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 z vašeho systému. Další informace viz téma "Konfigurace správce front pro přijetí připojení klienta na platformě Multiplatforms" na stránce 1034.

JMS 2.0 *Použití ukázkových aplikací IBM MQ classes for JMS*

Ukázkové aplikace IBM MQ classes for JMS poskytují přehled obecných funkcí rozhraní API produktu JMS . Můžete je použít k ověření instalace a nastavení serveru systému zpráv a k usnadnění vytváření vlastních aplikací.

Informace o této úloze

Potřebujete-li pomoc při vytváření vlastních aplikací, můžete ukázkové aplikace použít jako výchozí bod. Pro každou aplikaci je k dispozici jak zdrojová, tak i zkompilovaná verze. Zkontrolujte ukázkový zdrojový kód a identifikujte klíčové kroky pro vytvoření každého požadovaného objektu pro vaši aplikaci (ConnectionFactory, Connection, Session, Destination a Producent nebo Konzument nebo obojí) a nastavte všechny specifické vlastnosti, které jsou zapotřebí k určení způsobu, jakým má vaše aplikace fungovat. Další informace viz téma "Psaní aplikací IBM MQ classes for JMS/Jakarta Messaging" na stránce 135. Ukázky mohou být v budoucích verzích produktu IBM MQzměněny.

Pro JMS 2.0, Tabulka 11 na stránce 116 zobrazuje, kde jsou na každé platformě nainstalovány ukázkové aplikace IBM MQ classes for JMS .

Poznámka:

JM 3.0 Pro IBM MQ classes for Jakarta Messagingse připravují nové ukázky.

Tabulka 11. Instalační adresáře pro ukázkové aplikace IBM MQ classes for JMS

Platforma	**Adresář**
AIX AIX 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 existují podadresáře, které obsahují jednu nebo více ukázkových aplikací, jak ukazuje Tabulka 12 na stránce 116.

Tabulka 12. IBM MQ classes for JMS ukázkové aplikace

Název vzorku	**Popis**
JmsBrowser.java	Aplikace prohlížeče front systému JMS , která vyhledává všechny dostupné zprávy v uvedené frontě, aniž by je odebrali, v pořadí, v jakém by je přijímala aplikace spotřebitele.
JmsConsumer.java	Aplikace prohlížeče front systému JMS , která vyhledává všechny dostupné zprávy v uvedené frontě bez jejich odebrání v pořadí, v jakém by byly přijaty aplikací spotřebitele, vyhledáním instance továrny připojení a cílové instance v počátečním kontextu (tato ukázka podporuje pouze kontext systému souborů).
JmsJndiConsumer.java	Aplikace spotřebitele 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 výrobce JMS (odesilatel nebo vydavatel), která odesílá jednoduchou zprávu do uvedeného místa určení (fronta nebo téma) vyhledáním instance továrny připojení a cílové instance v počátečním kontextu (tato ukázka podporuje pouze kontext systému souborů).
JmsProducer.java	Aplikace výrobce JMS (odesilatel nebo vydavatel), která odesílá jednoduchou zprávu do uvedeného místa určení (fronta nebo téma).
/interactive/	
SampleConsumerJava.java	Příjem zpráv z tématu/fronty.
SampleProducerJava.java	Odeslat zprávu (zprávy) do tématu/fronty.
/interactive/helper/	
BaseOptions.java	Abstraktní třída, kterou lze rozšířit, aby poskytovala funkčnost uživatelských voleb.

Tabulka 12. IBM MQ classes for JMS ukázkové aplikace (pokračování)	
Název vzorku	**Popis**
IsValidType.java	Abstraktní třída pro třídy kontroly platnosti.
JmsApp.java	Abstraktní třída, kterou lze rozšířit, aby poskytovala funkčnost spotřebitele/producenta.
Keys.java	Sada klíčů, která definuje volby pro ukázkové aplikace.
Literals.java	Sada konstantních literálů.
MyContext.java	Kontext, ve kterém jsou volby prezentovány.
Options.java	Poskytuje funkčnost pro volby uživatele.
OptionsPresenter.java	Kontext, ve kterém jsou prezentovány aktuální volby.
/simple/	
SimpleAsyncPutPTP.java	Jednoduchá aplikace pro systém zpráv typu point-to-point; zpráva je odeslána asynchronně (také známá jako systém zpráv *fire-and-forget*). Nebyly přijaty žádné zprávy.
SimpleDurableSub.java	Jednoduchá aplikace, která demonstruje prostředek trvalého odběru.
SimpleJNDILookup.java	Minimální a jednoduchá aplikace, která demonstruje vyhledávání objektů JMS pomocí počátečního kontextu. Není vytvořeno žádné připojení ke správci front a nejsou odesílány ani přijímány žádné zprávy.
SimpleMQMDRead.java	Jednoduchá aplikace, která demonstruje, jak může aplikace JMS využít pole deskriptoru zpráv (MQMD) produktu MQ jako vlastnosti zprávy JMS . Nejsou odesílány žádné zprávy; předpokládá se, že používaná fronta je naplněna některými zprávami.
SimpleMQMDWrite.java	Jednoduchá aplikace, která demonstruje, jak může aplikace JMS psát pole deskriptoru zpráv produktu MQ (MQMD). Nebyly přijaty žádné zprávy.
SimplePTP.java	Minimální a jednoduchá aplikace pro systém zpráv typu point-to-point.
SimplePubSub.java	Minimální a jednoduchá aplikace pro systém zpráv publikování-odběr.
SimpleReadAheadPTP.java	Jednoduchá aplikace pro systém zpráv typu point-to-point. Zprávy jsou streamovány ze správce front (známého také jako prostředek dopředného čtení). Nejsou odesílány žádné zprávy; předpokládá se, že používaná fronta je naplněna některými zprávami.
SimpleRequestor.java	Jednoduchá aplikace, která používá žadatele k odeslání zprávy požadavku a poté čeká na odpověď a přijme ji. Poznámka: Předpokládá se, že některá jiná aplikace zpracuje zprávu požadavku a odešle zprávu odpovědi.
SimpleResponder.java	Jednoduchá aplikace, která naslouchá na místě určení zprávy a poté odešle odpověď na místo určení replyTo zprávy. Aplikace je napsána tak, aby fungovala ve spojení s ukázkou SimpleRequestor .
SimpleRetainedPub.java	Jednoduchá aplikace, která demonstruje zachované publikování. Nebyly přijaty žádné zprávy.

Tabulka 12. IBM MQ classes for JMS ukázkové aplikace (pokračování)	
Název vzorku	**Popis**
SimpleWMQJMSPTP.java	Minimální a jednoduchá aplikace pro systém zpráv typu point-to-point.
SimpleWMQJMSPubSub.java	Minimální a jednoduchá aplikace pro systém zpráv publikování/odběru.

IBM MQ classes for JMS poskytuje skript s názvem `runjms` , který lze použít ke spuštění ukázkových aplikací. Tento skript nastaví prostředí IBM MQ tak, aby vám umožňovalo spouštět ukázkové aplikace IBM MQ classes for JMS .

Tabulka 13 na stránce 118 zobrazuje umístění skriptu na každé platformě:

Tabulka 13. Umístění skriptu `runjms`	
Platforma	**Adresář**
AIX AIX Linux Linux	*MQ_INSTALLATION_PATH*`/java/bin/runjms`
Windows Windows	*MQ_INSTALLATION_PATH*`\java\bin\runjms.bat`

IBM i IBM i	`/qibm/proddata/mqm/java/bin/runjms`
	, nebo
	`/qibm/proddata/mqm/java/bin/runjms64`

z/OS z/OS	*MQ_INSTALLATION_PATH*`/java/bin/runjms`

Chcete-li použít skript `runjms` k vyvolání ukázkové aplikace, postupujte takto:

Postup

1. Otevřete 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. Zadáním následujícího příkazu spusťte ukázku s těmito parametry:

   ```
   Path to the runjms script/runjms sample_application_name parameters
   ```

Příklad

Linux Chcete-li například spustit ukázku `JmsBrowser` na 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 instalováno pro produkt IBM MQ classes for JMS” na stránce 85
Při instalaci produktu IBM MQ classes for JMSse vytvoří několik souborů a adresářů. V systému Windowsse některá konfigurace provádí během instalace automatickým nastavením proměnných prostředí. Na jiných platformách a v určitých prostředích Windows musíte před spuštěním aplikací IBM MQ classes for JMS nastavit proměnné prostředí.

Skripty dodávané s IBM MQ classes for JMS/Jakarta Messaging

K dispozici je řada skriptů, které pomáhají s běžnými úlohami, které je třeba provést při použití produktu IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging.

Tabulka 14 na stránce 119 vypisuje všechny skripty a jejich použití. Skripty jsou v podadresáři `bin` instalačního adresáře produktu IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging .

Tabulka 14. Skripty dodávané s produktem IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging

Utility	**Použijte**
Vyčištění [1]	Tento skript je udržován z důvodu kompatibility s předchozími verzemi, ale neprovádí žádnou funkci. Ruční vyčištění informací o odběru již není nutné.
DefaultConfiguration	Spustí aplikaci výchozí konfigurace na jiných platformách než Windows.
formatLog [1]	Tento skript je udržován z důvodu kompatibility s předchozími verzemi, ale neprovádí žádnou funkci. Výstup protokolu je nyní vytvořen v čitelném textu.
IVTRun [1] IVTSetup [1] IVTTidy [1]	Používá se v ověřovacím testu dvoubodové instalace, jak je popsáno v tématu “Dvoubodový IVT pro IBM MQ classes for JMS” na stránce 108.
JM 3.0 JMS30Admin [1]	Spustí nástroj pro administraci produktu IBM MQ Jakarta Messaging , jak je popsáno v tématu Spuštění nástroje pro administraci.
JM 3.0 JMS30Admin.config	Konfigurační soubor pro nástroj pro administraci produktu IBM MQ Jakarta Messaging , jak je popsáno v tématu Konfigurace nástroje pro administraci JMS.
JMS 2.0 JMSAdmin [1]	Spustí nástroj pro administraci produktu IBM MQ JMS , jak je popsáno v tématu Spuštění nástroje pro administraci.
JMS 2.0 JMSAdmin.config	Konfigurační soubor pro nástroj pro administraci produktu IBM MQ JMS , jak je popsáno v tématu Konfigurace nástroje pro administraci JMS.
PSIVTRun [1]	Spustí program pro ověření instalace publikování/odběru, jak je popsáno v tématu “IVT publikování/odběru pro IBM MQ classes for JMS” na stránce 112.
PSReportDump.třída	Tato třída je udržována z důvodu kompatibility s předchozími verzemi, ale neprovádí žádnou funkci.

Tabulka 14. Skripty dodávané s produktem IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging (pokračování)	
Utility	**Použijte**
V 9.3.0 JM 3.0 V 9.3.0 setjms30env “2” na stránce 120	Linux AIX Pro systém Jakarta Messaging 3.0nastaví proměnné prostředí pro spuštění aplikace IBM MQ classes for JMS ve 32bitovém prostředí JVM (Java Virtual Machine) na systémech AIX and Linux , jak je popsáno v tématu “Nastavení proměnných prostředí pro IBM MQ classes for JMS/Jakarta Messaging” na stránce 90.
JMS 2.0 setjmsenv “2” na stránce 120	Linux AIX Pro systém JMS 2.0nastaví proměnné prostředí pro spuštění aplikace IBM MQ classes for JMS ve 32bitovém prostředí JVM (Java Virtual Machine) na systémech AIX and Linux , jak je popsáno v tématu “Nastavení proměnných prostředí pro IBM MQ classes for JMS/Jakarta Messaging” na stránce 90.
V 9.3.0 JM 3.0 V 9.3.0 setjms30env64 “2” na stránce 120	Linux AIX Pro systém Jakarta Messaging 3.0nastaví proměnné prostředí pro spuštění aplikace IBM MQ classes for JMS v 64bitovém prostředí JVM na systémech AIX and Linux , jak je popsáno v tématu “Nastavení proměnných prostředí pro IBM MQ classes for JMS/Jakarta Messaging” na stránce 90.
JMS 2.0 setjmsenv64 “2” na stránce 120	Linux AIX Pro systém JMS 2.0nastaví proměnné prostředí pro spuštění aplikace IBM MQ classes for JMS v 64bitovém prostředí JVM na systémech AIX and Linux , jak je popsáno v tématu “Nastavení proměnných prostředí pro IBM MQ classes for JMS/Jakarta Messaging” na stránce 90.

Poznámka:

1. V systému Windowsmá název souboru příponu `.bat`.
2. Tyto skripty jsou k dispozici pouze v systému AIX and Linux . V systému Windowspo instalaci produktu IBM MQspusťte příkaz **setmqenv**. Další informace viz téma “Nastavení proměnných prostředí pro IBM MQ classes for JMS/Jakarta Messaging” na stránce 90.

Podpora pro OSGi s IBM MQ classes for JMS

OSGi poskytuje rámec, který podporuje implementaci aplikací jako svazků balíků. Svazky balíků OSGi jsou dodávány jako součást IBM MQ classes for JMS.

IBM MQ classes for JMS obsahuje následující svazky balíků OSGi.

com.ibm.msg.client.osgi.jms*version_number*.jar
: Společná vrstva kódu v souboru IBM MQ classes for JMS. Informace o vrstvené architektuře tříd IBM MQ pro systém JMSnaleznete v tématu IBM MQ pro architekturu JMS.

com.ibm.msg.client.osgi.jms.prereq_*version_number*.jar
: Předem požadované soubory JAR (Java archive) pro společnou vrstvu.

com.ibm.msg.client.osgi.commonservices.j2se_*version_number*.jar
: Obecné služby pro aplikace Java Platform, Standard Edition (Java SE).

com.ibm.msg.client.osgi.nls_*version_number*.jar
: Zprávy pro společnou vrstvu.

com.ibm.msg.client.osgi.wmq_*version_number*.jar
: Poskytovatel systému zpráv IBM MQ v adresáři IBM MQ classes for JMS. Informace o vrstvené architektuře produktu IBM MQ classes for JMSnaleznete v tématu IBM MQ pro architekturu JMS.

com.ibm.msg.client.osgi.wmq.prereq_*version_number*.jar
Nezbytné soubory JAR pro poskytovatele systému zpráv IBM MQ .

com.ibm.msg.client.osgi.wmq.nls_*version_number*.jar
Zprávy pro poskytovatele systému zpráv IBM MQ .

com.ibm.mq.jakarta.osgi.allclient_*version_number*.jar
V 9.3.0 JM 3.0 V 9.3.0 V případě systému Jakarta Messaging 3.0tento soubor JAR umožňuje aplikacím používat soubory IBM MQ classes for JMS i IBM MQ classes for Javaa také zahrnuje kód pro zpracování zpráv PCF.

com.ibm.mq.jakarta.osgi.allclientprereqs_*version_number*.jar
V 9.3.0 JM 3.0 V 9.3.0 Pro systém Jakarta Messaging 3.0tento soubor JAR poskytuje nezbytné předpoklady pro `com.ibm.mq.jakarta.osgi.allclient_version_number.jar`.

com.ibm.mq.osgi.allclient_*version_number*.jar
JMS 2.0 V případě systému JMS 2.0tento soubor JAR umožňuje aplikacím používat soubory IBM MQ classes for JMS i IBM MQ classes for Javaa také zahrnuje kód pro zpracování zpráv PCF.

com.ibm.mq.osgi.allclientprereqs_*version_number*.jar
JMS 2.0 Pro systém JMS 2.0tento soubor JAR poskytuje nezbytné předpoklady pro `com.ibm.mq.osgi.allclient_version_number.jar`.

kde *version_number* je číslo verze produktu IBM MQ , který je nainstalován.

Balíky jsou nainstalovány do podadresáře `java/lib/OSGi` vaší instalace produktu IBM MQ nebo do 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. Použití těchto svazků balíků odebere omezení, že nelze spustit IBM MQ classes for JMS i IBM MQ classes for Java v rámci stejného rámce OSGi, všechna ostatní omezení však stále platí.

Balík `com.ibm.mq.osgi.java`*version_number*`.jar`, který je také nainstalován do podadresáře `java/lib/OSGi` vaší instalace produktu IBM MQ , nebo složka `java\lib\OSGi` na systému Windows, je součástí adresáře IBM MQ classes for Java. Tento svazek balíků nesmí být načten do běhového prostředí OSGi, které má načtený soubor IBM MQ classes for JMS .

Svazky balíků OSGi pro IBM MQ classes for JMS byly zapsány do specifikace OSGi Release 4. Nefungují v prostředí OSGi Release 3.

Musíte správně nastavit cestu k systému nebo cestu ke knihovně, aby běhové prostředí OSGi mohlo najít všechny požadované soubory DLL nebo sdílené knihovny.

Pokud použijete svazky balíků OSGi pro IBM MQ classes for JMS, dočasná témata nefungují. Kromě toho nejsou třídy uživatelských procedur kanálu napsané v produktu Java podporovány kvůli vnitřnímu problému při načítání tříd v prostředí s více zavaděči tříd, například OSGi. Uživatelský balík si může být vědom balíků IBM MQ classes for JMS , ale balíky IBM MQ classes for JMS si nejsou vědomy žádného uživatelského balíku. V důsledku toho nemůže zavaděč tříd použitý v balíku IBM MQ classes for JMS načíst třídu uživatelské procedury kanálu, která je v uživatelském balíku.

Další informace o OSGi viz web OSGi Alliance .

z/OS MQ Adv. VUE JMS/Jakarta Messaging připojitelnost klienta k dávkovým aplikacím spuštěným na z/OS

Za určitých podmínek se aplikace IBM MQ classes for JMS/Jakarta Messaging v systému z/OS může připojit ke správci front v systému z/OS pomocí připojení klienta. Použití připojení klienta může zjednodušit topologie IBM MQ .

Pokud je aplikace IBM MQ classes for JMS/Jakarta Messaging spuštěna v dávkovém prostředí a použije se jedna z následujících podmínek, může se pomocí připojení klienta připojit ke vzdálenému správci front z/OS :

- LTS V 9.3.4 Kód IBM MQ classes for JMS/Jakarta Messaging je IBM MQ 9.3.4 nebo novější, nebo Long Term Support s použitou opravou APAR PH56722 . Správce front může mít libovolnou podporovanou verzi.
- Připojovaný správce front je spuštěn s oprávněním IBM MQ Advanced for z/OS Value Unit Edition , a proto má parametr **ADVCAP** nastaven na hodnotu ENABLED. Správce front může mít libovolnou podporovanou verzi.

Další informace o IBM MQ Advanced for z/OS Value Unit Edition viz IBM MQ identifikátory produktů a informace o exportu.

See ZOBRAZENÍ SPRÁVCE FRONT for more information on **ADVCAP** and Začátek QMGR for more information on **QMGRPROD**.

Všimněte si, že dávka je jediné podporované prostředí; neexistuje žádná podpora pro JMS/Jakarta Messaging pro CICS nebo JMS/Jakarta Messaging pro IMS.

Aplikace IBM MQ classes for JMS/Jakarta Messaging v systému z/OS nemůže používat připojení v režimu klienta pro připojení ke správci front, který není spuštěn v systému z/OS.

Pokud se aplikace IBM MQ classes for JMS/Jakarta Messaging v systému z/OS pokusí připojit pomocí režimu klienta a není to povoleno, zobrazí se zpráva o výjimce JMSFMQ0005 .

Podpora Advanced Message Security (AMS)

Aplikace IBM MQ classes for JMS/Jakarta Messaging klienta mohou používat produkt AMS při připojování ke vzdáleným správcům front z/OS v souladu s podmínkami dříve popsanými v tomto tématu.

Chcete-li použít produkt AMS tímto způsobem, musí aplikace klienta použít typ úložiště klíčů `jceracfks` v adresáři `keystore.conf`, kde:

- Předpona názvu vlastnosti je `jceracfks` a tato předpona názvu nerozlišuje velká a malá písmena.
- Úložiště klíčů je svazek klíčů RACF .
- Hesla nejsou vyžadována a budou ignorována. Důvodem je, že svazky klíčů RACF nepoužívají hesla.
- Pokud zadáte poskytovatele, poskytovatel musí být `IBMJCE`.

Používáte-li produkt `jceracfks` s produktem AMS, musí být úložiště klíčů ve formátu: `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 rozlišuje velikost písmen.

Následující příklad používá standardní AMS svazek klíčů pro uživatele `JOHNDOE`:

```
jceracfks.keystore=safkeyring://JOHNDOE/drq.ams.keyring
```

Související pojmy

“Java připojitelnost klienta k dávkovým aplikacím spuštěným na z/OS” na stránce 359
Za určitých podmínek se aplikace IBM MQ classes for Java v systému z/OS může připojit ke správci front v systému z/OS pomocí připojení klienta. Použití připojení klienta může zjednodušit topologie IBM MQ .

Získání IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging samostatně

Knihovny a obslužné programy nezbytné pro spouštění aplikací pomocí produktu IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging jsou k dispozici v samorozbalujícím souboru JAR,

který stáhnete z webu Fix Central. To provedete, pokud chcete získat pouze tyto soubory, například pro implementaci do nástroje pro správu softwaru nebo pro použití se samostatnými klientskými aplikacemi.

Než začnete

Před spuštěním této úlohy se ujistěte, že máte v počítači nainstalováno prostředí Java runtime environment (JRE) a že bylo prostředí JRE přidáno do systémové cesty.

Instalační program produktu Java , který se používá v tomto procesu instalace, nevyžaduje spuštění jako uživatel root ani žádný konkrétní uživatel. Jediným požadavkem je, aby uživatel, pod kterým se spouští, měl přístup pro zápis do adresáře, do kterého chcete soubory vjet.

V 9.3.0 JM 3.0 V 9.3.0 Od IBM MQ 9.3.0, Jakarta Messaging 3.0 je podporován pro vývoj nových aplikací. Produkt IBM MQ 9.3.0 nadále podporuje produkt JMS 2.0 pro existující aplikace. Použití rozhraní API Jakarta Messaging 3.0 a rozhraní API JMS 2.0 ve stejné aplikaci není podporováno. Další informace naleznete v tématu Použití tříd IBM MQ pro systém zpráv JMS/Jakarta.

Informace o této úloze

Samorozbalovací soubor JAR se používá k minimalizaci velikosti stahování a času, který trvá provedení extrakce. Přesný obsah tohoto souboru JAR a podadresářů, do kterých extrahuje soubory, závisí na verzi souboru IBM MQ.

Když spustíte samorozbalovací soubor JAR, zobrazí se licenční smlouva IBM MQ , která musí být přijata. Také vám umožňuje změnit nadřízený adresář pro extrakci.

V případě souboru IBM MQ 9.3extrahuje samorozbalovací soubor JAR soubory do následující adresářové struktury:

wmq/JavaEE
Soubory EAR a RAR adaptéru prostředků IBM MQ .

JMS 2.0 Následující soubory jsou určeny pro použití s objekty JMS 2.0 a JMS 1.1 :

- `wmq.jmsra.ivt.ear`
- `wmq.jmsra.rar`

JM 3.0 Existuje také ekvivalentní sada pro použití s objekty Jakarta Messaging 3.0 :

- `wmq.jakarta.jmsra.ivt.ear`. Obsahuje soubory testu ověření instalace.
- `wmq.jakarta.jmsra.rar`. Obsahuje soubory adaptéru prostředků.

wmq/JavaSE

wmq/JavaSE/bin
Nástroje **JMSAdmin** a **JMS30Admin** . Používá se k definování entit JNDI , které představují objekty JMS nebo Jakarta Messaging .

JMS 2.0 Následující soubory jsou určeny pro použití s objekty JMS 2.0 a JMS 1.1 :

- `JMSAdmin.bat`
- `JMSAdmin`
- `JMSAdmin.config`

JM 3.0 Existuje také ekvivalentní sada pro použití s objekty Jakarta Messaging 3.0 :

- `JMS30Admin.bat`. Soubor, který se používá ke spuštění nástroje v systému Windows.
- `JMS30Admin`. Skript, který se používá ke spuštění nástroje na platformách Linux a UNIX .
- `JMS30Admin.config`. Ukázkový konfigurační soubor pro nástroj.

Poznámka:

- Před přidáním nástroje **JMSAdmin** do samorozbalovacího souboru JAR byly soubory v tomto adresáři v nadřízeném adresáři `wmq/JavaSE`.
- Klient, který je nainstalován pomocí samorozbalovacího souboru JAR, může použít nástroj **JMSAdmin** nebo **JMS30Admin** k vytvoření spravovaných objektů systému zpráv Java v rámci kontextu systému souborů (soubor `.bindings`). Klient může také vyhledat a používat tyto spravované objekty.
- JMS 2.0 Nástroj **JMSAdmin** pro použití s objekty JMS 2.0 a JMS 1.1 byl přidán do samorozbalovacího souboru JAR v adresáři IBM MQ 9.2.0 Fix Pack 2 a IBM MQ 9.2.2.
- JM 3.0 Nástroj **JMS30Admin** pro použití s objekty Jakarta Messaging 3.0 byl přidán do samorozbalovacího souboru JAR v umístění IBM MQ 9.3.0.

wmq/JavaSE/lib

Rozšířené zabezpečení zpráv používá následující balíky Bouncy Castle s otevřeným zdrojem pro podporu syntaxe kryptografických zpráv (CMS). Viz Podpora jiných prostředí JRE nežIBM s AMS.

V 9.3.5 Pro Continuous Delivery z IBM MQ 9.3.5:

- `bcpkix-jdk18on.jar`
- `bcprov-jdk18on.jar`
- `bcutil-jdk18on.jar`

LTS Pro Long Term Support a Continuous Delivery před IBM MQ 9.3.5:

- `bcpkix-jdk15on.jar`
- `bcprov-jdk15on.jar`
- `bcutil-jdk15on.jar`

Následující soubory obsahují třídy pro specifickou úroveň JMS nebo Jakarta Messaging :

- JMS 2.0 `com.ibm.mq.allclient.jar` (JMS 2.0 a JMS 1.1)
- JM 3.0 `com.ibm.mq.jakarta.client.jar` (Jakarta Messaging 3.0)

Další nezbytné soubory JAR:

- V 9.3.3 Removed `com.ibm.mq.traceControl.jar`. Používá se k dynamickému řízení trasování pro aplikace IBM MQ classes for JMS .
- `fscontext.jar`. Nezbytné, pokud vaše aplikace provádí vyhledávání rozhraní JNDI pomocí kontextu systému souborů.
- V 9.3.3 `jackson-annotations.jar`, `jackson-core.jar`, `jackson-databind.jar`: Obsahuje třídy, které se používají k provádění mapování CipherSuite a CipherSpec při vytváření zabezpečených připojení TLS ke správci front.
- JM 3.0 `jakarta.jms-api.jar`. Obsahuje rozhraní Jakarta Messaging 3.0 a definice výjimek.
- JMS 2.0 `jms.jar`. Obsahuje rozhraní JMS 2.0 a definice výjimek.
- `org.json.jar`. Obsahuje třídy, které umožňují produktu IBM MQ classes for JMS interpretovat soubory CCDT ve formátu JSON.
- `providerutil.jar`. Nezbytné, pokud vaše aplikace provádí vyhledávání rozhraní JNDI pomocí kontextu systému souborů.

Poznámka: Stabilized Soubory `com.ibm.mq.allclient.jar` a `com.ibm.mq.jakarta.client.jar` obsahují kopii souboru IBM MQ classes for Java. V produktu IBM MQ 9.0jsou však tyto třídy deklarovány jako funkčně stabilizované na úrovni dodané v produktu IBM MQ 8.0. Viz Zamítnutí, stabilizace a odebrání na adrese IBM MQ 9.0.

wmq/OSGi
Balíky klienta OSGi IBM MQ :

- JM 3.0 com.ibm.mq.jakarta.osgi.allclient_*V.R.M.F*.jar
- JM 3.0 com.ibm.mq.jakarta.osgi.allclientprereqs_*V.R.M.F*.jar
- JMS 2.0 com.ibm.mq.osgi.allclient_*V.R.M.F*.jar
- JMS 2.0 com.ibm.mq.osgi.allclientprereqs_*V.R.M.F*.jar

kde *V.R.M.F* je číslo verze, vydání, úpravy a opravné sady.

Postup

1. Stáhněte soubor JAR klienta IBM MQ Java / JMS z adresáře Fix Central.
   a) Klepněte na tento odkaz: IBM MQ Java / JMS klient.
   b) Vyhledejte klienta pro svou verzi produktu IBM MQ v zobrazeném seznamu dostupných oprav.
   Příklad:

   ```
   release level: 9.3.0.0-IBM-MQ-Install-Java-All
   Long Term Support: 9.3.0.0 IBM MQ JMS and Java 'All Client'
   ```

   Poté klepněte na název klientského souboru a postupujte podle procesu stahování.
2. Spusťte extrakci z adresáře, do kterého jste stáhli soubor.
   Chcete-li spustit extrakci, zadejte příkaz v následujícím formátu:

   ```
   java -jar V.R.M.F-IBM-MQ-Install-Java-All.jar
   ```

   kde *V.R.M.F* je číslo verze produktu, například `9.3.0.0`, a *V.R.M.F*`-IBM-MQ-Install-Java-All.jar` je název souboru, který byl stažen z adresáře Fix Central.
   Chcete-li například extrahovat klienta JMS pro vydání IBM MQ 9.3.0 , použijte následující příkaz:

   ```
   java -jar 9.3.0.0-IBM-MQ-Install-Java-All.jar
   ```

   Poznámka: Chcete-li provést tuto instalaci, musíte mít na svém počítači nainstalováno prostředí JRE a přidáno do systémové cesty.

   Když zadáte příkaz, zobrazí se následující informace:

   ```
   Než budete moci použít, extrahovat nebo nainstalovat produkt IBM MQ V9.3, musíte přijmout
   podmínky 1. IBM Mezinárodní licenční smlouva pro hodnocení
   Programy 2. IBM Mezinárodní licenční smlouva pro programy a další
   informace o licenci. Přečtěte si pozorně následující licenční smlouvy.

   Licenční smlouvu lze samostatně zobrazit pomocí
   --viewLicenseVolba smlouvy.

   Stiskněte klávesu Enter, chcete-li nyní zobrazit licenční podmínky, nebo 'x', chcete-li
   přeskočit.
   ```
3. Přezkoumejte a přijměte licenční podmínky:
   a) Chcete-li zobrazit licenci, stiskněte klávesu Enter.
   Případně stiskněte tlačítko x pro přeskočení zobrazení licence.
   Po zobrazení licence nebo okamžitě po stisknutí tlačítka x se zobrazí následující zpráva:

   ```
   Další informace o licenci lze samostatně zobrazit pomocí
   --viewLicenseVolba informací.

   Stiskněte klávesu Enter, chcete-li nyní zobrazit další informace o licenci, nebo 'x',
   chcete-li přeskočit.
   ```

   b) Chcete-li zobrazit další licenční podmínky, stiskněte klávesu Enter.
   Případně stiskněte tlačítko x, chcete-li přeskočit zobrazení dalších licenčních podmínek.

Po zobrazení dalších licenčních podmínek nebo okamžitě, pokud stisknete tlačítko x, se zobrazí následující zpráva:

```
Výběrem možnosti "Souhlasím" níže souhlasíte s podmínkami
licenční smlouva a jiné podmínky nežIBM , jsou-li použitelné. Pokud ne,
souhlasím, vyberte "nesouhlasím".

Vyberte [ 1] Souhlasím, nebo [ 2] NeSouhlasím:
```

c) Chcete-li přijmout licenční smlouvu a pokračovat ve výběru instalačního adresáře, vyberte 1.

Případně vyberte 2 pro okamžité ukončení instalace.

Vyberete-li volbu 1, zobrazí se zpráva podobná následující zprávě:

```
Zadejte adresář pro soubory produktu nebo ponechte pole prázdné, chcete-li přijmout výchozí
hodnotu.
Výchozí cílový adresář je H: \downloads

Cílový adresář pro soubory produktu?
```

4. Zadejte nadřízený adresář pro extrakci.

Výchozí umístění je aktuální adresář.

- Chcete-li extrahovat soubory produktu do výchozího umístění, stiskněte klávesu Enter bez uvedení hodnoty.
- Chcete-li extrahovat soubory produktu do jiného umístění, zadejte název adresáře, do kterého chcete extrahovat soubory, a stisknutím klávesy Enter spusťte extrakci.

Zadaný název adresáře již nesmí existovat, jinak při spuštění extrakce dojde k chybě a nejsou nainstalovány žádné soubory.

Za předpokladu, že již neexistuje, je vytvořen uvedený adresář a programové soubory jsou extrahovány do tohoto adresáře. Během instalace se vytvoří nový adresář s názvem `wmq` v rámci nadřízeného adresáře, který jste zadali.

V adresáři `wmq` se vytvoří tři podadresáře `JavaEE`, `JavaSE`a `OSGis` následujícím obsahem:

JavaEE

JM 3.0 `wmq.jakarta.jmsra.ivt.ear`

JM 3.0 `wmq.jakarta.jmsra.rar`

JMS 2.0 `wmq.jmsra.ivt.ear`

JMS 2.0 `wmq.jmsra.rar`

JavaSE

Tento adresář obsahuje následující podadresáře a soubory:

JavaSE/lib

V 9.3.5 `bcpkix-jdk18on.jar`

LTS `bcpkix-jdk15on.jar`

V 9.3.5 `bcprov-jdk18on.jar`

LTS `bcprov-jdk15on.jar`

V 9.3.5 `bcutil-jdk18on.jar`

LTS `bcutil-jdk15on.jar`

JMS 2.0 `com.ibm.mq.allclient.jar`

JM 3.0 `com.ibm.mq.jakarta.client.jar`

V 9.3.3 Removed `com.ibm.mq.traceControl.jar`

`fscontext.jar`

V 9.3.3 jackson-annotations.jar

V 9.3.3 jackson-core.jar

V 9.3.3 jackson-databind.jar

jms.jar

org.json.jar

providerutil.jar

JavaSE/bin

JMSAdmin.bat

JMSAdmin

JMSAdmin.config

Specifikace OSGi

JM 3.0 com.ibm.mq.jakarta.osgi.allclient_*V.R.M.F*.jar

JM 3.0 com.ibm.mq.jakarta.osgi.allclientprereqs_*V.R.M.F*.jar

JMS 2.0 com.ibm.mq.osgi.allclient_*V.R.M.F*.jar

JMS 2.0 com.ibm.mq.osgi.allclientprereqs_*V.R.M.F*.jar

Po dokončení extrakce se zobrazí potvrzovací zpráva, jak ukazuje následující příklad:

```
Extrahování souborů do H: \downloads\wmq
Všechny soubory produktu se úspěšně extrahovaly.
```

Povolení v IBM MQ classes for JMS/Jakarta Messaging

Mechanizmus serializace a deserializace objektů Java byl identifikován jako potenciální bezpečnostní riziko. Povolení v adresáři IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging poskytují určitou ochranu před některými riziky serializace.

Informace o této úloze

Mechanizmus serializace a deserializace objektů Java byl identifikován jako potenciální bezpečnostní riziko, protože deserializace vytváří instance libovolných objektů Java , kde existuje potenciál pro zlomyslně odeslaná data způsobující různé problémy. Jedna z významných aplikací serializace je v Jakarta Messaging 3.0 a Java Message Service 2.0 ObjectMessages , které používají serializaci k zapouzdření a přenosu libovolných objektů.

Seznam povolení serializace je potenciálním zmírněním některých rizik, která serializace představuje. Tím, že explicitně určíte, které třídy mohou být zapouzdřeny a extrahovány z ObjectMessages, poskytuje příkaz allowlisting určitou ochranu před některými riziky serializace.

Související pojmy

“Spuštění aplikací IBM MQ classes for JMS pod Java security manager” na stránce 103
IBM MQ classes for JMS lze spustit s povoleným Java security manager . Chcete-li úspěšně spustit aplikace s povoleným produktem Java security manager , musíte nakonfigurovat prostředí Java Virtual Machine (JVM) pomocí vhodného konfiguračního souboru zásad.

Koncepce seznamu povolení

V systémech IBM MQ classes for JMS a IBM MQ classes for Jakarta Messagingexistuje podpora pro povolení výpisu tříd v implementaci rozhraní JMS ObjectMessage . To poskytuje potenciální zmírnění některých bezpečnostních rizik, která potenciálně souvisejí s mechanizmem serializace a deserializace objektů Java .

Seznam povolených položek v adresáři IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging

Důležité:

Termín *allowlist* nahradil termín *whitelist*. V případě produktu IBM MQ 9.0 a novějších verzí to zahrnuje některé názvy systémových vlastností Java uvedené v tomto tématu. Nemusíte měnit žádnou existující konfiguraci. Předchozí názvy systémové vlastnosti budou také pracovat.

IBM MQ classes for JMS (JMS 2.0) V 9.3.0 V 9.3.0 a IBM MQ classes for Jakarta Messaging (Jakarta Messaging 3.0) podporují funkci allowlisting tříd v implementaci rozhraní JMS ObjectMessage .

- JMS 2.0 Pro systém IBM MQ classes for JMSjsou relevantní názvy vlastností **com.ibm.mq.jms.allowlist.***.
- JM 3.0 Pro systém IBM MQ classes for Jakarta Messagingjsou relevantní názvy vlastností **com.ibm.mq.jakarta.jms.allowlist.***

Seznam povolení definuje, které třídy Java mohou být serializovány pomocí ObjectMessage.setObject() a deserializovány pomocí ObjectMessage.getObject().

- JMS 2.0 Pokusy o serializaci nebo deserializaci instance třídy, která není zahrnuta v seznamu povolených položek s položkou ObjectMessage , způsobí vyvolání výjimky javax.jms.MessageFormatException s příčinou java.io.InvalidClassException .
- JM 3.0 Pokusy o serializaci nebo deserializaci instance třídy, která není zahrnuta v seznamu povolení s volbou ObjectMessage , způsobí vyvolání výjimky jakarta.jms.MessageFormatException s příčinou java.io.InvalidClassException .

Vytvoření seznamu povolení

Důležité: Položky IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging nelze distribuovat se seznamem povolených položek. Volba tříd, které se mají přenést pomocí ObjectMessages , je volbou návrhu aplikace a IBM MQ tomu nemůže zabránit.

Z tohoto důvodu mechanismus povolování umožňuje dva režimy provozu:

Zjišťování
V tomto režimu mechanizmus vytvoří výpis úplných názvů tříd, přičemž ohlásí všechny třídy, u kterých bylo zjištěno, že byly serializovány nebo deserializovány v části ObjectMessages.

Vynucení
V tomto režimu mechanizmus vynucuje povolení výpisu a odmítne pokusy o serializaci nebo deserializaci tříd, které nejsou v seznamu povolených.

Chcete-li použít tento mechanismus, musíte nejprve spustit v režimu DISCOVERY, abyste shromáždili seznam aktuálně serializovaných a deserializovaných tříd, přezkoumat seznam a použít jej jako základ pro váš seznam povolení. Dokonce by mohlo být vhodné používat seznam beze změny, ale seznam musí být nejprve přezkoumán, než se rozhodnete tak učinit.

Řízení mechanizmu povolování

Pro řízení mechanismu povolování jsou k dispozici tři systémové vlastnosti:

com.ibm.mq.jms.allowlist (JMS 2.0) a com.ibm.mq.jakarta.jms.allowlist (Jakarta Messaging 3.0)
Tuto vlastnost lze zadat jedním z následujících způsobů:

- Název cesty k souboru, který obsahuje seznam povolení, ve formátu identifikátoru URI souboru (tj. začínající na `file:`). V režimu DISCOVERY se do tohoto souboru zapisuje mechanizmem allowlisting. Soubor nesmí existovat. Pokud soubor existuje, mechanizmus vygeneruje výjimku a nepřepíše jej. V režimu ENFORCEMENT je tento soubor přečten mechanizmem allowlisting.
- Čárkami oddělené úplné názvy tříd, které tvoří seznam povolených položek.

Není-li tato vlastnost nastavena, mechanizmus seznamu povolení je neaktivní.

Používáte-li soubor Java security manager, musíte se ujistit, že soubory JAR IBM MQ classes for JMS mají k tomuto souboru přístup pro čtení a zápis.

com.ibm.mq.jms.allowlist.discover (JMS 2.0) a com.ibm.mq.jakarta.jms.allowlist.discover (Jakarta Messaging 3.0)

- Je-li tato vlastnost nenastavena nebo nastavena na hodnotu false, mechanizmus seznamu povolení se spustí v režimu ENFORCEMENT.
- Je-li tato vlastnost nastavena na hodnotu true a seznam povolení byl uveden jako identifikátor URI souboru, mechanizmus seznamu povolení se spustí v režimu DISCOVERY.
- Je-li tato vlastnost nastavena na hodnotu true a seznam povolení byl zadán jako seznam názvů tříd, mechanizmus seznamu povolení vygeneruje vhodnou výjimku.
- Je-li tato vlastnost nastavena na hodnotu true a seznam povolení nebyl zadán pomocí parametru Vlastnost com.ibm.mq.jms.allowlist nebo com.ibm.mq.jakarta.jms.allowlist, mechanizmus seznamu povolení je neaktivní.
- Pokud je tato vlastnost nastavena na hodnotu true a soubor allowlist již existuje, mechanizmus allowlist vygeneruje výjimku java.io.InvalidClassException a položky se do souboru nepřidají.

com.ibm.mq.jms.allowlist.mode (JMS 2.0) a com.ibm.mq.jakarta.jms.allowlist.mode (Jakarta Messaging 3.0)

Tuto vlastnost řetězce lze zadat jedním ze tří způsobů:

- Je-li tato vlastnost nastavena na hodnotu SERIALIZE, režim ENFORCEMENT provede ověření seznamu povolení pouze na metodě ObjectMessage.setObject().
- Je-li tato vlastnost nastavena na hodnotu DESERIALIZE, režim ENFORCEMENT provede ověření seznamu povolení pouze na metodě ObjectMessage.getObject().
- Pokud není tato vlastnost nastavena nebo je nastavena na jinou hodnotu, pak režim ENFORCEMENT provede ověření seznamu povolení na obou metodách ObjectMessage.getObject() a ObjectMessage.setObject().

Formát souboru seznamu povolení

Toto jsou hlavní vlastnosti formátu souboru allowlist:

- Soubor allowlist je ve výchozím kódování souboru platformy s řádkovými koncovkami vhodnými pro platformu.

  Poznámka: Pokud se používá soubor allowlist, pak se tento soubor vždy zapíše a přečte pomocí výchozího kódování souboru pro prostředí JVM.

  To je v pořádku, pokud je soubor allowlist generován některým z následujících způsobů:

  – z/OS Generováno samostatnou aplikací spuštěnou na systému z/OS a používanou jinými samostatnými aplikacemi, které jsou také spuštěny na systému z/OS.

  – Generováno aplikací spuštěnou uvnitř produktu WebSphere Application Server na libovolné platformě a používané jinou instancí produktu WebSphere Application Server.

  – Multi Generováno samostatnou aplikací spuštěnou v systému IBM MQ for Multiplatformsa používanou jinými samostatnými aplikacemi spuštěnými v systému IBM MQ for Multiplatformsnebo aplikacemi spuštěnými v produktu WebSphere Application Server na libovolné platformě.

  Protože však produkt WebSphere Application Server používá kód ASCII a samostatné prostředí JVM používá kód EBCDIC, dojde k problémům s kódováním souborů, pokud je soubor seznamu povolení generován jedním z následujících způsobů:

  – Generováno v systému z/OS, poté používáno samostatnými aplikacemi spuštěnými na jiné platformě než z/OS nebo produktem WebSphere Application Server.

  - Generováno buď produktem WebSphere Application Server , nebo samostatnou aplikací spuštěnou na jiné platformě než z/OS, poté používáno samostatnou aplikací v systému z/OS.
- Každý neprázdný řádek obsahuje úplný název třídy. Prázdné řádky jsou ignorovány.
- Komentáře mohou být zahrnuty-cokoli za znakem '#', až na konec řádku, je ignorováno.
- Existuje velmi základní mechanismus pro divokou zvěř:
  - '*' může být **posledním** prvkem názvu třídy.
  - '*' odpovídá **jednomu** prvku názvu třídy, tj. třídě, ale žádné části balíku.

  Takže `com.ibm.mq.*` by se shodovalo s `com.ibm.mq.MQMessage` , ale ne `com.ibm.mq.jmqi.remote.api.RemoteFAP`.

  Zástupné znaky nefungují pro třídy ve výchozím balíku, který je pro třídy bez explicitního názvu balíku, takže název třídy "*" je odmítnut.
- Nesprávně formátované soubory seznamu povolení, například soubory obsahující položku, jako je `com.ibm.mq.*.Message`, kde zástupný znak není posledním prvkem, způsobí výjimku java.lang.IllegalArgumentException .
- Prázdný soubor allowlist má za následek úplné zakázání použití ObjectMessage.

Formát seznamu povolení jako seznam oddělený čárkami

Pro seznam povolení je k dispozici stejný mechanizmus vytváření zástupných znaků jako seznam oddělený čárkami.

- Znak '*' může být rozšířen operačním systémem, pokud je zadán na příkazovém řádku nebo ve skriptu shellu nebo v dávkovém souboru, takže může vyžadovat zvláštní zacházení.
- Znak komentáře '#' lze použít pouze v případě, že je zadán soubor. Pokud je seznam povolení uveden jako seznam názvů tříd oddělených čárkami, pak za předpokladu, že jej operační systém nebo shell nezpracovává, protože se jedná o výchozí znak komentáře v mnoha shellech AIX and Linux , je považován za normální znak.

Kdy se povolení stane?

Příkaz allowlisting se zahájí, když aplikace poprvé spustí metodu ObjectMessage setMessage() nebo getMessage().

Vyhodnotí se systémové vlastnosti, otevře se soubor allowlist a v režimu ENFORCEMENT se při inicializaci mechanizmu načte seznam povolených tříd. V tomto bodě se položka zapíše do souboru protokolu produktu IBM MQ JMS pro aplikaci.

Při inicializaci mechanizmu nemusí být jeho parametry změněny. Vzhledem k tomu, že čas inicializace není snadno předpovězen, protože závisí na chování aplikace. Nastavení systémových vlastností a obsah souboru seznamu povolení by proto měly být považovány za pevné od okamžiku spuštění aplikace. Neměňte vlastnosti ani obsah souboru allowlist za běhu aplikace, protože výsledky nejsou zaručeny.

Body, které je třeba zvážit

Nejlepším přístupem ke zmírnění rizik, která jsou podstatná pro mechanizmus serializace Java , by bylo prozkoumat alternativní přístupy k přenosu dat, jako např. použití JSON namísto ObjectMessage. Pomocí mechanizmů Advanced Message Security (AMS) můžete přidat další zabezpečení tím, že zajistíte, že zprávy budou pocházet z důvěryhodných zdrojů.

Pokud s aplikací používáte mechanizmus Java security manager , musíte udělit následující oprávnění:

- FilePermission na libovolném souboru seznamu povolení, který používáte, s oprávněním ke čtení pro režim ENFORCEMENT, oprávnění k zápisu pro režim DISCOVER.
- JMS 2.0 PropertyPermission (číst) na vlastnostech **`com.ibm.mq.jms.allowlist`**, **`com.ibm.mq.jms.allowlist.discover`**a **`com.ibm.mq.jms.allowlist.mode`** .

- JM 3.0 PropertyPermission (číst) na vlastnostech **com.ibm.mq.jakarta.jms.allowlist**, **com.ibm.mq.jakarta.jms.allowlist.discover**a **com.ibm.mq.jakarta.jms.allowlist.mode** .

Další informace

Další informace o seznamech povolení viz “Nastavení a použití seznamu povolení JMS nebo Jakarta Messaging” na stránce 131 a “Seznam povolených položek v adresáři WebSphere Application Server” na stránce 133 .

Související pojmy

“Spuštění aplikací IBM MQ classes for JMS pod Java security manager” na stránce 103
IBM MQ classes for JMS lze spustit s povoleným Java security manager . Chcete-li úspěšně spustit aplikace s povoleným produktem Java security manager , musíte nakonfigurovat prostředí Java Virtual Machine (JVM) pomocí vhodného konfiguračního souboru zásad.

Nastavení a použití seznamu povolení JMS nebo Jakarta Messaging

Tyto informace informují o tom, jak seznam povolení funguje a jak jej nastavíte pomocí funkcí obsažených v souboru IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging pro generování souboru seznamu povolení, který obsahuje seznam typů ObjectMessages , které může aplikace zpracovat.

Než začnete

Důležité:

Termín *allowlist* nahradil termín *whitelist*. V případě produktu IBM MQ 9.0 a novějších verzí to zahrnuje některé názvy systémových vlastností Java uvedené v tomto tématu. Nemusíte měnit žádnou existující konfiguraci. Předchozí názvy systémové vlastnosti budou také pracovat.

Před spuštěním této úlohy se ujistěte, že jste přečetli a porozuměli “Koncepce seznamu povolení” na stránce 127

Informace o této úloze

Protože JMS a Jakarta Messaging sdílejí mnoho společného, lze další odkazy na JMS v tomto tématu považovat za odkazy na oboje. Případné rozdíly jsou podle potřeby zvýrazněny.

Pokud jste povolili funkci allowlisting, produkt IBM MQ classes for JMS tuto funkčnost používá následujícími způsoby:

- Když chce aplikace odeslat zprávu ObjectMessage, může ji vytvořit jedním ze dvou způsobů voláním:
  - Session.createObjectMessage(Serializable), předávající objekt, který má být obsažen ve zprávě.
  - Session.createObjectMessage(), chcete-li vytvořit prázdnou ObjectMessagea poté volat ObjectMessage.setObject(Serializable) pro uložení objektu, který má být odeslán uvnitř ObjectMessage.

  Jsou-li volány metody Session.createObjectMessage(Serializable) nebo ObjectMessage.setObject(Serializable) (Serializable), třídy pro službu JMS kontrolují, zda předaný objekt je typu, který je uveden v seznamu povolených.

  Pokud se jedná o uvedený typ, objekt je serializován a uložen v rámci ObjectMessage. Pokud je však objekt typu, který není v seznamu povolených, IBM MQ classes for JMS vygeneruje výjimku JMSException obsahující zprávu:

  ```
  JMSCC0052: Při serializaci objektu došlo k výjimce:
  'java.io.InvalidClassException: < třída objekt>; Třída nemusí být serializována
  nebo deserializováno, protože nebylo zahrnuto do seznamu povolení '< allowlist>'.
  ```

  zpět k aplikaci.

Důležité: Dojde-li k výjimce z metody Session.createObjectMessage(Serializable), zpráva ObjectMessage nebude vytvořena. Podobně, pokud je vyvolána výjimka JMSException z metody ObjectMessage.setObject(Serializable), objekt nebude přidán do ObjectMessage.

- Pokud aplikace obdrží zprávu ObjectMessage, zavolá metodu ObjectMessage.getObject(), aby získala objekt, který je v ní obsažen. Když je tato metoda volána, IBM MQ classes for JMS zkontrolujte typ objektu obsaženého v ObjectMessage, abyste zjistili, zda je tento objekt typu uvedeného v seznamu povolených.

  Pokud ano, je objekt deserializován a vrácen do aplikace. Pokud je však objekt typu, který není v seznamu povolených, IBM MQ classes for JMS vygeneruje výjimku JMSException obsahující zprávu:

  ```
  JMSCC0053: Při deserializaci zprávy došlo k výjimce:
  'java.io.InvalidClassException: < třída objekt>; třída nesmí být
  serializováno nebo deserializováno, protože nebylo zahrnuto do
  allowlist '< allowlist>'. '.
  ```

  zpět k aplikaci.

Předpokládejme například, že vaše aplikace obsahuje následující kód pro odeslání zprávy ObjectMessage obsahující 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);
```

Vzhledem k tomu, že není povolen výpis povolení, může aplikace zprávu úspěšně vložit 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 seznamu povolení je povolena. Aplikace je stále schopna vytvořit a odeslat zprávu ObjectMessage obsahující objekt typu java.net.URI , protože tento typ je uveden v seznamu povolení.

Pokud však změníte soubor `allowlist.txt` tak, aby obsahoval jedinou položku `java.util.Calendar`, bude při volání aplikace funkce allowlist stále povolena:

```
ObjectMessage msg = session.createObjectMessage(testURL);
```

IBM MQ classes for JMS zkontrolujte seznam povolení a zjistěte, že neobsahuje položku pro java.net.URI.

V důsledku toho dojde k výjimce 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) {
      :    :    :    :    :    :    :
```

Není-li povolen výpis povolení, aplikace může přijímat zprávy ObjectMessages , které obsahují objekt libovolného typu. Aplikace poté před provedením příslušného zpracování zkontroluje, zda je objekt typu java.net.URL .

Pokud nyní spustíte aplikaci se systémovou vlastností Java :

```
-Dcom.ibm.mq.jms.allowlist=java.net.URL
```

, funkce seznamu povolení 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() vygeneruje výjimku JMSException obsahující zprávu JMSCC0053 . Aplikace se poté musí rozhodnout, co se zprávou provede. Zpráva může být například přesunuta do fronty nedoručených zpráv pro daného správce front.

Aplikace se vrátí normálně pouze v případě, že objekt uvnitř ObjectMessage je typu java.net.URL.

Postup

1. Spusťte aplikaci, která zpracovává ObjectMessages, s následujícími vlastnostmi systému Java :

   ```
   -Dcom.ibm.mq.jms.allowlist.discover=true
   -Dcom.ibm.mq.jms.allowlist=file:/<path to your allowlist file>
   ```

   Při spuštění aplikace produkt IBM MQ classes for JMS vytvoří soubor obsahující typy objektů, které aplikace zpracovala.
2. Po zpracování reprezentativního vzorku aplikace ObjectMessages v určitém časovém úseku ji zastavte.

   Soubor allowlist nyní obsahuje seznam všech typů objektů obsažených v ObjectMessages , které aplikace zpracovala za běhu.

   Pokud jste aplikaci spustili dostatečně dlouhou dobu, tento seznam obsahuje všechny možné typy objektů obsažených v ObjectMessages , které aplikace pravděpodobně zpracuje.
3. Restartujte aplikaci s následující sadou systémových vlastností:

   ```
   -Dcom.ibm.mq.jms.allowlist=file:/<path to your allowlist file>
   ```

   To povolí výpis povolení, a pokud IBM MQ classes for JMS zjistí zprávu ObjectMessage typu, který není v seznamu povolených, dojde k výjimce JMSException obsahující buď zprávu JMSCC0052 , nebo JMSCC0053 .

Seznam povolených položek v adresáři WebSphere Application Server

Způsob použití příkazu IBM MQ classes for JMS allowlisting in WebSphere Application Server.

Důležité:

Termín *allowlist* nahradil termín *whitelist*. V případě produktu IBM MQ 9.0 a novějších verzí to zahrnuje některé názvy systémových vlastností Java uvedené v tomto tématu. 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.

Další informace o používání těchto dvou produktů naleznete v části “Společné použití IBM MQ a WebSphere Application Server” na stránce 485 .

Funkce IBM MQ 9.0.0 Fix Pack 1 dále zahrnují příslušné funkce.

Po aktualizaci aplikačního serveru můžete použít systémové vlastnosti Java :

- `-Dcom.ibm.mq.jms.allowlist`
- `-Dcom.ibm.mq.jms.allowlist.discover`

popsané v části “Nastavení a použití seznamu povolení JMS nebo Jakarta Messaging” na stránce 131.

Poznámka: Musíte nastavit systémové vlastnosti Java jako generické argumenty prostředí JVM na serveru Java Virtual Machine , který se používá ke spuštění aplikačního serveru, a restartovat aplikační server, aby se změny projevily.

Další informace naleznete v části *Generické argumenty prostředí JVM* v části Java Nastavení virtuálního počítače .

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 povolení *youruserId_MyObject*. Aplikační server zpracovává pouze objekty tohoto typu.

Následující nastavení:

```
-Dcom.ibm.mq.jms.allowlist.discover=true
-Dcom.ibm.mq.jms.allowlist=file:C/:allowlist.txt
```

konfigurovat aplikační server pro použití režimu *Zjistit* a zaznamenat podrobnosti o JMS ObjectMessages, 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.txta` použije informace v tomto souboru k určení seznamu povolení.

Související pojmy

“Spuštění aplikací IBM MQ classes for JMS pod Java security manager” na stránce 103
IBM MQ classes for JMS lze spustit s povoleným Java security manager . Chcete-li úspěšně spustit aplikace s povoleným produktem Java security manager , musíte nakonfigurovat prostředí Java Virtual Machine (JVM) pomocí vhodného konfiguračního souboru zásad.

Převody znakových řetězců v souboru IBM MQ classes for JMS

IBM MQ classes for JMS používají CharsetEncoders a CharsetDecoders přímo pro převod znakových řetězců. Výchozí chování pro převod znakových řetězců lze konfigurovat se dvěma systémovými vlastnostmi. Zpracování zpráv, které obsahují nemapovatelné znaky, lze konfigurovat pomocí vlastností zprávy pro nastavení akce UnmappableCharactera nahrazujících bajtů.

Před IBM MQ 8.0byly převody řetězců v souboru IBM MQ classes for JMS provedeny voláním metod `java.nio.charset.Charset.decode(ByteBuffer)` a `Charset.encode(CharBuffer)` .

Použití jedné z těchto metod má za následek výchozí náhradu (`REPLACE`) chybných nebo nepřeložitelných dat. Toto chování může zakrývat chyby v aplikacích a vést k neočekávaným znakům, například `?`, v přeložených datech.

V systému IBM MQ 8.0lze tyto problémy zjišťovat dříve a efektivněji pomocí IBM MQ classes for JMS přímo používat CharsetEncoders a CharsetDecoders a explicitně konfigurovat zpracování chybných a nepřeložitelných dat. Výchozí chování je `REPORT` takové problémy vyvoláním vhodného `MQException`.

Konfigurace

Překlad z UTF-16 (reprezentace znaků používaná v produktu Java) do nativní znakové sady, například UTF-8, se nazývá *kódování*, zatímco překlad v opačném směru se nazývá *dekódování*.

Dekódování má výchozí chování pro `CharsetDecodersa` hlásí chyby vyvoláním výjimky.

Jedno nastavení se používá k určení `java.nio.charset.CodingErrorAction` pro řízení ošetřování chyb při kódování i dekódování. Jedno další nastavení se používá k řízení náhradního bajtu nebo bajtů při kódování. V operacích dekódování bude použit výchozí řetězec náhrady Java .

UnmappableCharacterNastavení akce a bajtů náhrady v IBM MQ classes for JMS

V produktu IBM MQ 8.0jsou k dispozici následující dvě vlastnosti pro nastavení akce UnmappableCharactera nahrazující bajty. Příslušné definice konstant jsou uvedeny v souboru `com.ibm.msg.client.wmq.WMQConstants`.

JMS_IBM_UNMAPPABLE_ACTION
Nastaví nebo získá `CodingErrorAction` , který se má použít, když znak nelze mapovat v operaci kódování nebo dekódování.

Tuto hodnotu byste měli nastavit takto: `CodingErrorAction.{REPLACE|REPORT|IGNORE}.toString()` :

```
public static final String JMS_IBM_UNMAPPABLE_ACTION = "JMS_IBM_Unmappable_Action";
```

JMS_IBM_UNMAPPABLE_REPLACEMENT
Nastaví nebo získá náhradní bajty, které se mají použít, když znak nelze mapovat v operaci kódování.

Výchozí řetězec náhrady Java se používá v operacích dekódování.

```
public static final String JMS_IBM_UNMAPPABLE_REPLACEMENT = "JMS_IBM_Unmappable_Replacement";
```

Vlastnosti `JMS_IBM_UNMAPPABLE_ACTION` a `JMS_IBM_UNMAPPABLE_REPLACEMENT` lze nastavit pro cíle nebo zprávy. Hodnota nastavená na zprávě přepíše hodnotu nastavenou na místě určení, kam se zpráva odesílá.

Všimněte si, že parametr `JMS_IBM_UNMAPPABLE_REPLACEMENT` musí být nastaven jako jeden bajt.

Systémové vlastnosti pro nastavení předvoleb systému

V produktu IBM MQ 8.0jsou k dispozici následující dvě systémové vlastnosti Java pro konfiguraci výchozího chování, pokud jde o převod znakového řetězce.

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 získá náhradní bajty, které se mají použít, když znak nelze mapovat v operaci kódování. Výchozí řetězec náhrady Java se používá v operacích dekódování.

Chcete-li se vyhnout záměně znakových a nativních bajtových reprezentací Java , měli byste zadat `com.ibm.mq.cfg.jmqi.UnmappableCharacterReplacement` jako desetinné číslo představující náhradní bajt v nativní znakové sadě.

Například desetinná hodnota ?, jako nativní bajt, je 63, pokud je nativní znaková sada založená na ASCII, jako např. ISO-8859-1, zatímco 111, pokud je nativní znaková sada EBCDIC.

Poznámka: Všimněte si, že pokud má objekt MQMD nebo MQMessage nastavena pole **unmappableAction** nebo **unMappableReplacement** , mají hodnoty těchto polí přednost před systémovými vlastnostmi Java . To umožňuje v případě potřeby přepsat hodnoty zadané systémovými vlastnostmi Java pro každou zprávu.

Související pojmy
“Převody znakových řetězců v souboru IBM MQ classes for Java” na stránce 339
IBM MQ classes for Java používají CharsetEncoders a CharsetDecoders přímo pro převod znakových řetězců. Výchozí chování pro převod znakových řetězců lze konfigurovat se dvěma systémovými vlastnostmi. Zpracování zpráv, které obsahují nemapovatelné znaky, lze konfigurovat prostřednictvím produktu `com.ibm.mq.MQMD`.

Psaní aplikací IBM MQ classes for JMS/Jakarta Messaging

Po stručném úvodu k modelu JMS tato sekce poskytuje podrobné pokyny, jak psát aplikace IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging .

Informace o této úloze

V 9.3.0 JM 3.0 V 9.3.0 Od IBM MQ 9.3.0, Jakarta Messaging 3.0 je podporován pro vývoj nových aplikací. Produkt IBM MQ 9.3.0 nadále podporuje produkt JMS 2.0 pro existující aplikace.

Použití rozhraní API Jakarta Messaging 3.0 a rozhraní API JMS 2.0 ve stejné aplikaci není podporováno. Další informace naleznete v tématu Použití tříd IBM MQ pro systém zpráv JMS/Jakarta.

Související pojmy

IBM MQ classes for Jakarta Messaging: přehled

Model JMS a Jakarta Messaging

Modely JMS a Jakarta Messaging definují sadu rozhraní, která mohou aplikace Java používat k provádění operací systému zpráv. IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging jsou oba poskytovatelé systému zpráv. Definují, jak objekty JMS a Jakarta Messaging souvisí s koncepcemi IBM MQ . Specifikace JMS a Jakarta Messaging očekávají, že určité objekty JMS a Jakarta Messaging budou spravovány.

JMS 2.0 Produkt IBM MQ 8.0 přidal podporu pro verzi JMS 2.0 standardu JMS , který zavedl zjednodušené rozhraní API a zároveň zachoval klasické rozhraní API z produktu JMS 1.1.

V 9.3.0 JM 3.0 V 9.3.0 V produktu IBM MQ 9.3.0je produkt Jakarta Messaging 3.0 podporován pro vývoj nových aplikací. Produkt IBM MQ 9.3.0 nadále podporuje produkt JMS 2.0 pro existující aplikace. Použití rozhraní API JMS 2.0 a rozhraní API Jakarta Messaging 3.0 ve stejné aplikaci není podporováno.

Poznámka: V případě systému Jakarta Messaging 3.0se řízení specifikace JMS přesune z Oracle do Java Community Process. Produkt Oracle si však zachovává řízení názvu "javax", který se používá v jiných technologiích Java , jež nebyly přesunuty do procesu komunity Java . Takže zatímco Jakarta Messaging 3.0 je funkčně ekvivalentní produktu JMS 2.0 , existují určité rozdíly v pojmenování:

- Oficiální název pro verzi 3.0 je Jakarta Messaging spíše než Java Message Service.
- Názvy balíků a konstant mají předponu `jakarta` spíše než `javax`. Například v produktu JMS 2.0 je počátečním připojením k poskytovateli systému zpráv objekt `javax.jms.Connection` a v produktu Jakarta Messaging 3.0 objekt `jakarta.jms.Connection` .

JMS 2.0 Balíky javax.jms definují rozhraní 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.

JM 3.0 Balíky jakarta.jms definují rozhraní Jakarta Messaging a poskytovatel Jakarta Messaging implementuje tato rozhraní pro specifický produkt systému zpráv. IBM MQ classes for Jakarta Messaging je poskytovatel Jakarta Messaging , který implementuje rozhraní Jakarta Messaging pro IBM MQ.

Protože JMS a Jakarta Messaging sdílejí mnoho společného, lze další odkazy na JMS v tomto tématu považovat za odkazy na oboje. Případné rozdíly jsou podle potřeby zvýrazněny.

Zjednodušené rozhraní API

Produkt JMS 2.0 zavedl zjednodušené rozhraní API a zároveň zachoval rozhraní specifická pro doménu a nezávislá na doméně z produktu JMS 1.1. Zjednodušené rozhraní API snižuje počet objektů, které jsou potřebné pro odesílání a příjem zpráv, a skládá se z následujících rozhraní:

ConnectionFactory
ConnectionFactory je spravovaný objekt, který používá klient JMS k vytvoření připojení. Toto rozhraní se také používá v klasickém rozhraní API.

JMSKontext
Tento objekt kombinuje objekty připojení a relace klasického rozhraní API. JMSKontextové objekty lze vytvářet z jiných objektů JMSKontext, přičemž základní připojení je duplikováno.

JMSProducent
Producent JMSje vytvořen kontextem JMSa používá se k odesílání zpráv do fronty nebo tématu. Objekt Producent JMSzpůsobí vytvoření objektů, které jsou nezbytné pro odeslání zprávy.

JMSSpotřebitel
Konzument JMSje vytvořen kontextem JMSa používá se k přijímání zpráv z tématu nebo fronty.

Zjednodušené rozhraní API má řadu efektů:

- Objekt kontextu JMSvždy automaticky spustí základní připojení.
- JMSProducenti a JMSSpotřebitelé mohou nyní pracovat přímo s těly zpráv, aniž by museli získat celý objekt zprávy, pomocí metody getBody zprávy.
- Vlastnosti zprávy lze nastavit na objektu Producent JMSpomocí řetězení metod před odesláním 'těla' obsahu zprávy. Producent JMSzpracuje vytvoření všech objektů, které jsou potřebné k odeslání zprávy. Pomocí funkce 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 zavedl sdílené odběry, ve kterých mohou být zprávy sdíleny mezi více spotřebiteli. Se všemi odběry produktu JMS 1.1 se zachází jako s nesdílenými odběry.

Klasické rozhraní API

Následující seznam shrnuje hlavní rozhraní 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 vytváření relací.

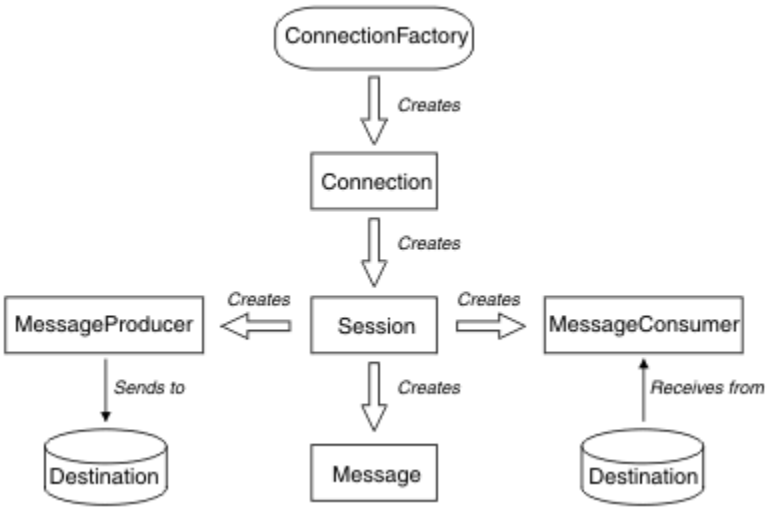
Relace
Relace je kontext s jedním podprocesem pro odesílání a příjem zpráv. Aplikace používá relaci k vytváření zpráv, producentů zpráv a spotřebitelů zpráv. Relace je buď transakční, nebo netransakční.

Zpráva
Objekt zprávy zapouzdřuje zprávu, kterou aplikace odesílá nebo přijímá.

MessageProducer
Aplikace používá producenta zpráv k odesílání zpráv do místa určení.

MessageConsumer
Aplikace používá spotřebitele zpráv k přijímání zpráv odeslaných do místa určení.

Obrázek 9 na stránce 138 zobrazuje tyto objekty a jejich vztahy.



Obrázek 9. Objekty JMS a jejich vztahy

Objekt Destination, ConnectionFactorynebo Connection může být souběžně používán různými podprocesy aplikace s podporou podprocesů, ale objekty Session, MessageProducernebo MessageConsumer nemohou být souběžně používány různými podprocesy. Nejjednodušším způsobem, jak zajistit, aby se objekt Session, MessageProducernebo MessageConsumer souběžně nepoužíval, je vytvořit pro každý podproces samostatný objekt relace.

Domény systému zpráv

Produkt JMS podporuje dva styly systému zpráv:

- Dvoubodový systém zpráv
- Systém zpráv publikování/odběru

Tyto styly systému zpráv jsou také označovány jako *domény systému zpráv*a v aplikaci můžete kombinovat oba styly systému zpráv. V doméně typu point-to-point je cílem fronta a v doméně publikování/odběru je cílem téma.

U verzí produktu JMS před JMS 1.1programování pro doménu typu point-to-point 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 použít společnou sadu rozhraní a metod, které podporují obě domény systému zpráv. Společná rozhraní poskytují zobrazení každé domény systému zpráv nezávislé na doméně. Tabulka 15 na stránce 138 vypíše JMS rozhraní nezávislá na doméně a jejich odpovídající rozhraní specifická pro doménu.

Tabulka 15. Rozhraní nezávislá na doméně JMS a specifická pro doménu

Rozhraní nezávislá na doméně	**Rozhraní specifická pro doménu pro doménu typu point-to-point**	**Specifická rozhraní domény 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

JMS 2.0 Produkt IBM MQ classes for JMS 2.0 podporuje jak dřívější rozhraní specifická pro doménu JMS 1.1 , tak zjednodušené rozhraní API produktu JMS 2.0. Produkt IBM MQ classes for JMS 2.0 lze proto použít pro údržbu existujících aplikací, včetně vývoje nových funkcí ve stávajících aplikacích.

JM 3.0 Produkt IBM MQ classes for Jakarta Messaging 3.0 podporuje verze Jakarta Messaging stejných rozhraní a doporučuje se pro vývoj nových aplikací.

V systémech IBM MQ classes for JMS a IBM MQ classes for Jakarta Messagingse objekty JMS vztahují k konceptům IBM MQ následujícími způsoby:

- Objekt připojení má vlastnosti odvozené od vlastností továrny připojení, která byla použita k vytvoření připojení. Tyto vlastnosti řídí způsob připojení aplikace ke správci front. Příklady těchto vlastností jsou název správce front a v případě aplikace, která se připojuje ke správci front v režimu klienta, název hostitele nebo adresa IP systému, v němž je správce front spuštěn.
- Objekt relace zapouzdřuje manipulátor připojení IBM MQ , který proto definuje transakční rozsah relace.
- Objekt MessageProducer a objekt MessageConsumer zapouzdřují popisovač objektu IBM MQ .

Při použití IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messagingplatí všechna normální pravidla IBM MQ . Všimněte si zejména, že aplikace může odeslat zprávu do vzdálené fronty, ale může přijmout zprávu pouze z fronty vlastněné správcem front, ke kterému je aplikace připojena.

Specifikace JMS očekává, že objekty ConnectionFactory a Destination budou spravovány. Administrátor vytvoří a udržuje spravované objekty v centrálním úložišti a aplikace JMS načte tyto objekty pomocí rozhraní JNDI (Java Naming and Directory Interface).

V produktu IBM MQ classes for JMS a IBM MQ classes for Jakarta Messagingje implementace rozhraní Destination abstraktní supertřída Queue a Topic, takže instance Destination je buď objekt Queue, nebo objekt Topic. Rozhraní nezávislá na doméně považují frontu nebo téma za cíl. Doména systému zpráv pro objekt MessageProducer nebo MessageConsumer je určena podle toho, zda je cílem fronta nebo téma.

V operačním systému IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging proto mohou být objekty následujících typů spravovány:

- ConnectionFactory
- Továrna QueueConnection
- Továrna TopicConnection
- Fronta
- Téma
- XAConnectionFactory
- Továrna XAQueueConnection
- Továrna XATopicConnection

Související pojmy

Jazyková rozhraní produktu IBM MQ Java

“Vytvoření a konfigurace továren připojení a míst určení” na stránce 198

Aplikace IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging může vytvářet továrny připojení a místa určení jejich načtením jako spravovaných objektů z oboru názvů rozhraní JNDI (Java Naming and Directory Interface) pomocí rozšíření IBM JMS nebo pomocí rozšíření IBM MQ JMS . Aplikace může také použít rozšíření produktu IBM JMS nebo rozšíření produktu IBM MQ JMS k nastavení vlastností továren připojení a míst určení.

Zprávy produktu JMS

Zprávy 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 vestavěný prostředek pro podporu hodnot vlastností definovaných aplikací. Vlastnosti poskytují účinný mechanismus pro filtrování zpráv definovaných aplikací.

Tělo
JMS definuje pět typů těla zprávy, které pokrývají většinu aktuálně používaných stylů systému zpráv:

Proud
Proud primitivních hodnot Java . Vyplňuje se a čte postupně.

Mapa
Sada dvojic název-hodnota, kde názvy jsou řetězce a hodnoty jsou primitivní typy Java .
K položkám lze přistupovat postupně nebo náhodně podle názvu. Pořadí položek není definováno.

Text
Zpráva obsahující java.lang.String.

Objekt
Zpráva, která obsahuje serializovatelný objekt Java .

Bajty
Proud neinterpretovaných bajtů. Tento typ zprávy je určen pro doslovné kódování těla zprávy tak, aby odpovídalo existujícímu formátu zprávy.

Pole záhlaví JMSCorrelationID se používá k propojení jedné zprávy s jinou. Obvykle propojuje zprávu odpovědi se zprávou, která ji požaduje. JMSCorrelationID může obsahovat ID zprávy specifické pro poskytovatele, řetězec specifický pro aplikaci nebo nativní bajt poskytovatele [].

Selektory zpráv v adresáři JMS
Zprávy mohou obsahovat hodnoty vlastností definované aplikací. Aplikace může použít selektory zpráv, aby měla zprávy filtru poskytovatele JMS .

Zpráva obsahuje vestavěný prostředek pro podporu hodnot vlastností definovaných aplikací. Ve skutečnosti to poskytuje mechanizmus pro přidání polí záhlaví specifických pro aplikaci do zprávy. Vlastnosti umožňují aplikaci, která používá selektory zpráv, nechat poskytovatele JMS vybrat nebo filtrovat zprávy jejím jménem pomocí kritérií specifických pro aplikaci. Vlastnosti definované aplikací musí splňovat následující pravidla:

- Názvy vlastností se musí řídit pravidly pro identifikátor selektoru zpráv.
- Hodnoty vlastností mohou být Boolean, byte, short, int, long, float, double a String.
- Předpony názvů JMSX a JMS_ jsou vyhrazeny.

Hodnoty vlastností jsou nastaveny před odesláním zprávy. Když klient obdrží zprávu, vlastnosti zprávy jsou jen pro čtení. Pokud se klient pokusí nastavit vlastnosti v tomto bodě, dojde k výjimce MessageNotWriteableException . Je-li volána volba clearProperties , lze vlastnosti nyní číst i do nich zapisovat.

Hodnota vlastnosti může duplikovat hodnotu v těle zprávy. Produkt JMS nedefinuje zásadu pro to, co se může stát vlastností. Vývojáři aplikací si však musí uvědomit, že poskytovatelé produktu JMS pravděpodobně zpracovávají data v těle zprávy efektivněji než data ve vlastnostech zprávy. Pro dosažení nejlepšího výkonu musí aplikace používat vlastnosti zpráv pouze v případě, že potřebují upravit záhlaví zprávy. Primární příčinou je podpora přizpůsobeného výběru zpráv.

Selektor zpráv JMS umožňuje klientovi určit zprávy, o které má zájem, pomocí záhlaví zprávy. Doručeny jsou pouze zprávy se záhlavími, která odpovídají selektoru.

Selektory zpráv nemohou odkazovat na hodnoty těla zprávy.

Selektor zpráv odpovídá zprávě, když se selektor vyhodnotí jako pravdivý, když pole záhlaví zprávy a hodnoty vlastností nahradí jejich odpovídající identifikátory v selektoru.

Selektor zpráv je řetězec se syntaxí, která je založena na podmnožině syntaxe podmíněného výrazu SQL92 . Pořadí, ve kterém je selektor zpráv vyhodnocován, je zleva doprava v rámci úrovně priority. Chcete-li změnit toto pořadí, můžete použít závorky. Předdefinované literály selektoru a názvy operátorů jsou zde psány velkými písmeny, ale nerozlišují velká a malá písmena.

Obsah selektoru zpráv

Selektor zpráv může obsahovat:

- literály
  - Řetězcový literál je uzavřen v uvozovkách. Dvojité uvozovky představují uvozovky. Příklady jsou 'literal' a 'literal'. Stejně jako řetězcové literály Java používají kódování znaků Unicode.
  - Přesný číselný literál je číselná hodnota bez desetinné čárky, například 57, -957 a +62. Podporována jsou čísla v rozsahu Java long.
  - Přibližný číselný literál je číselná hodnota ve vědecké notaci, například 7E3 nebo -57.9E2, nebo číselná hodnota s desetinným číslem, například 7., -95.7nebo +6.2. Čísla v rozsahu Java double jsou podporována.
  - Logické literály TRUE a FALSE.
- Identifikátory:
  - Identifikátor je posloupnost Java písmen a Java číslic s neomezenou délkou, přičemž první z nich musí být písmeno Java . Písmeno je libovolný znak, pro který metoda Character.isJavaLetter vrací hodnotu true. To zahrnuje _ a $. Písmeno nebo číslice je libovolný znak, pro který metoda Character.isJavaLetterOrDigit vrací hodnotu 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.

    JMSMessageID, JMSTimestamp, JMSCorrelationIDa JMSType mohou mít hodnotu null, a pokud ano, jsou považovány za hodnotu NULL.
  - Libovolný název začínající na JMSX je JMS-defined property name.
  - Libovolný název začínající na JMS_ je název vlastnosti specifický pro poskytovatele.
  - Jakýkoli název, který nezačíná na JMS , je název vlastnosti specifické pro aplikaci. Pokud existuje odkaz na vlastnost, která ve zprávě neexistuje, její hodnota je NULL. Pokud existuje, jeho hodnota je odpovídající hodnota vlastnosti.
- Mezera je stejná jako ta, která je definována pro Java: mezera, vodorovný tabelátor, posuv formuláře a ukončovač řádku.
- Výrazy:
  - Selektor je podmíněný výraz. Selektor, který se vyhodnotí jako pravdivý, se shoduje; selektor, který se vyhodnotí jako nepravdivý nebo neznámý, se neshoduje.
  - Aritmetické výrazy se skládají ze sebe, aritmetických operací, identifikátorů (s hodnotou, která je považována za číselný literál) a číselných literálů.
  - Podmíněné výrazy se skládají ze sebe samých, operací porovnání a logických operací.
- Standardní bracketing () pro nastavení pořadí, ve kterém jsou výrazy vyhodnocovány, je podporován.
- Logické operátory v pořadí podle pořadí: NOT, AND, OR.
- Operátory porovnání: =, >, > =, <, < =, < > (nerovná se).

  - Porovnávat lze pouze hodnoty stejného typu. Jednou z výjimek je, že je platné porovnat přesné číselné hodnoty a přibližné číselné hodnoty. (Nezbytný převod typu je definován pravidly číselného povýšení Java .) Pokud dojde k pokusu o porovnání různých typů, selektor bude mít vždy hodnotu false.
  - Porovnání řetězců a logických hodnot je omezeno na hodnoty = a < >. Dva řetězce se rovnají pouze v případě, že obsahují stejnou posloupnost znaků.
- Aritmetické operátory v pořadí podle priority:
  - +,-unární.
  - *,/, násobení a dělení.
  - +,-, sčítání a odčítání.
  - Aritmetické operace s hodnotou NULL nejsou podporovány. Pokud se o ně pokusíte, úplný selektor bude mít vždy hodnotu false.
  - Aritmetické operace musí používat číselné povýšení Java .
- arithmetic-expr1 [NOT] BETWEEN arithmetic-expr2 a arithmetic-expr3 porovnávací operátor:
  - Věk BETWEEN 15 and 19 je ekvivalentní věku > = 15 AND věk < = 19.
  - Věk NOT mezi 15 a 19 je ekvivalentní věku < 15 NEBO věku > 19.
  - Pokud má některý z výrazů operace BETWEEN hodnotu NULL, hodnota operace je false. Pokud je některý z výrazů operace NOT BETWEEN NULL, hodnota operace je true.
- identifikátor [NOT] IN (string-literal1, string-literal2, ...) operátor porovnání, kde identifikátor má hodnotu String nebo NULL.
  - Země IN ("Spojené království", "USA", "Francie") platí pro "Spojené království" a pro "Peru". Je ekvivalentní výrazu (Country = "UK") OR (Country = "US") OR (Country = "Francie").
  - Země NOT IN ("UK", "US", "France") je nepravdivá pro "UK" a pravdivá pro "Peru". Je ekvivalentní výrazu NOT ((Country = 'UK ') OR (Country = 'US') OR (Country = 'Francie ')).
  - Je-li identifikátor operace IN nebo NOT IN NULL, hodnota operace je neznámá.
- Identifikátor [NOT] LIKE vzorová hodnota [ESCAPE escape escape-character] porovnávací operátor, kde identifikátor má hodnotu řetězce. pattern-value je řetězcový literál, kde _ představuje libovolný jednotlivý znak a% představuje libovolnou posloupnost znaků (včetně prázdné posloupnosti). Všechny ostatní znaky stojí za sebou. Volitelný řídicí znak je jednoznakový řetězcový literál, se znakem, který se používá pro změnu významu speciálního významu _ a% v hodnotě vzorku.
  - phone LIKE '12%3' je true pro 123 a 12993 a false pro 1234.
  - slovo LIKE 'l_se' je true pro "lose" a false pro "loose".
  - podtržení 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.
- identifikátor IS NULL porovnávací operátor testuje hodnotu pole záhlaví null nebo chybějící hodnotu vlastnosti.
  - prop_name IS NULL.
- identifikátor IS NOT NULL porovnávací operátor testuje existenci nenulové hodnoty pole záhlaví nebo hodnoty vlastnosti.
  - Název položky NENÍ NULL.

Příklad selektoru zpráv

Následující selektor zpráv vybírá zprávy s typem zprávy auto, modrou barvou a hmotností větší než 2500 lbs:

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

Hodnoty vlastností NULL

Jak bylo uvedeno v předchozím seznamu, hodnoty vlastností mohou být NULL. Vyhodnocení výrazů selektoru, které obsahují hodnoty NULL, je definováno sémantikou SQL 92 NULL. V následujícím seznamu je uveden stručný popis těchto sémantik:

- SQL považuje hodnotu NULL za neznámou.
- Porovnání nebo aritmetika s neznámou hodnotou vždy vede k neznámé hodnotě.
- Operátor IS NULL převede neznámou hodnotu na hodnotu TRUE.
- Operátor IS NOT NULL převede neznámou hodnotu na hodnotu FALSE.

Speciální chování položek JMSMessageID a JMSCorrelationID

Třídy IBM MQ pro JMS obsahují optimalizace při výběru zpráv z fronty na základě JMSMessageID nebo JMSCorrelationID.

Pokud aplikace uvádí selektor formuláře:

JMSMessageID= 'ID:*ID_zprávy*'

kde *ID_zprávy* je řetězec obsahující standardní identifikátor zprávy IBM MQ , pak třídy IBM MQ pro JMS používají identifikátor **MatchOption** MQMO_MATCH_MSG_ID k získání zprávy s uvedeným identifikátorem zprávy.

Chcete-li například získat zprávu s identifikátorem zprávy 414D51207061756C745639314C545320C57C1A5F25ECE602 z fronty, měla by aplikace použít následující selektor zpráv:

```
JMSMessageID='ID:414D51207061756C745639314C545320C57C1A5F25ECE602'
```

Podobně, pokud aplikace uvádí selektor, který má formát:

JMSCorrelationID = 'ID:*correlation_id*'

kde *correlation_id* je řetězec obsahující standardní IBM MQ identifikátor korelace, třídy IBM MQ pro JMS používají k získání zprávy s určeným identifikátorem korelace z fronty identifikátor **MatchOption** MQMO_MATCH_CORREL_ID.

V následujícím příkladu se selektor zpráv používá k získání zprávy, která má identifikátor korelace 414D51207061756C745639314C545320846E5B5F25B1CC02:

```
JMSCorrelationID='ID:414D51207061756C745639314C545320846E5B5F25B1CC02'
```

Pokud selektor zpráv obsahuje hodnotu všech nul pro *message_id* nebo *correlation_id*, pak odpovídá jakékoli zprávě ve frontě. Pokud například aplikace používá selektor:

```
JMSMessageID='ID:000000000000000000000000000000000000000000000000'
```

pak každá zpráva ve frontě je považována za shodnou a vrácena do aplikace.

Další informace o MQMO_MATCH_MSG_ID a MQMO_MATCH_CORREL_ID **MatchOptions**naleznete v tématu MatchOptions (MQLONG).

Omezení

Ačkoli SQL podporuje pevné desetinné porovnání a aritmetiku, JMS selektory zpráv tak nečiní. To je důvod, proč jsou přesné číselné literály omezeny na ty, které nemají desetinné číslo. To je také důvod, proč existují číslice s desetinným číslem jako alternativní reprezentace pro přibližnou číselnou hodnotu.

Komentáře SQL nejsou podporovány.

Mapování zpráv JMS na zprávy IBM MQ
Zprávy IBM MQ se skládají z deskriptoru zprávy, volitelného záhlaví MQRFH2 a těla zprávy. Obsah zprávy JMS je částečně mapován a částečně zkopírován do zprávy IBM MQ .

Toto téma popisuje, jak je struktura zprávy JMS , která je popsána v první části této sekce, mapována na zprávu IBM MQ . Je zajímavé pro programátory, kteří chtějí přenášet zprávy mezi aplikacemi JMS a tradičními aplikacemi IBM MQ . Je také zajímavé pro osoby, které chtějí manipulovat se zprávami přenášenými mezi dvěma aplikacemi JMS , například v implementaci produktu IBM Integration Bus .

Tento oddíl se nepoužije, pokud aplikace používá připojení ke zprostředkovateli v reálném čase. Když aplikace používá připojení v reálném čase, veškerá komunikace se provádí přímo přes TCP/IP; nejsou zahrnuty žádné IBM MQ fronty nebo zprávy.

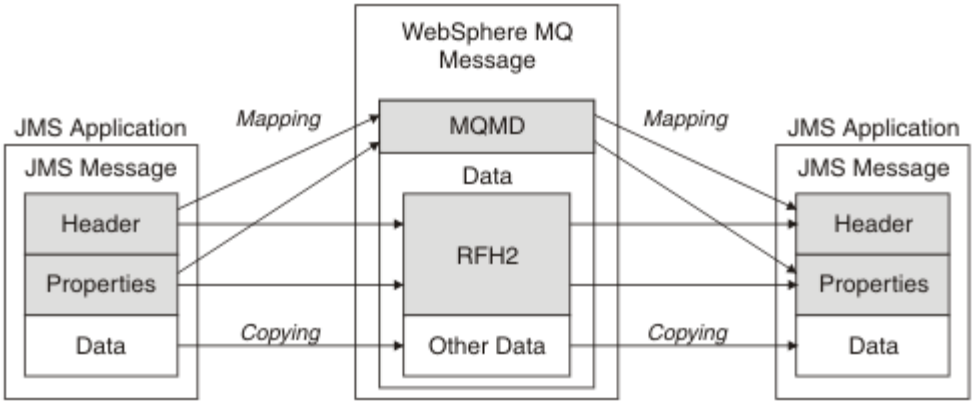
Zprávy IBM MQ se skládají ze tří komponent:

- Deskriptor zprávy IBM MQ (MQMD)
- Záhlaví IBM MQ MQRFH2
- Tělo zprávy.

MQRFH2 je volitelný a jeho zahrnutí do odchozí zprávy je řízeno příznakem TARGCLIENT ve třídě cíle JMS . Tento příznak můžete nastavit pomocí administračního nástroje produktu IBM MQ JMS . Vzhledem k tomu, že MQRFH2 obsahuje JMSspecifické informace, vždy je začleňte do zprávy, když odesilatel ví, že cílem příjmu je aplikace JMS . Při odesílání zprávy přímo do jiné aplikace nežJMS obvykle vynechte MQRFH2 . Důvodem je skutečnost, že taková aplikace ve zprávě produktu IBM MQ neočekává položku MQRFH2 .

Pokud příchozí zpráva nemá záhlaví MQRFH2 , objekt Queue nebo Topic odvozený z pole záhlaví JMSReplyTo zprávy má standardně nastaven tento příznak, takže zpráva odpovědi odeslaná do fronty nebo tématu také nemá záhlaví MQRFH2 . Toto chování zahrnutí záhlaví MQRFH2 do zprávy odpovědi můžete vypnout pouze v případě, že má původní zpráva záhlaví MQRFH2 , a to nastavením vlastnosti TARGCLIENTMATCHING továrny připojení na hodnotu NO.

Obrázek 10 na stránce 144 ukazuje, jak je struktura zprávy JMS transformována na zprávu IBM MQ a znovu:



Obrázek 10. Způsob transformace zpráv mezi produktem JMS a produktem IBM MQ pomocí záhlaví MQRFH2

Struktury jsou transformovány dvěma způsoby:

Mapování
Pokud modul MQMD obsahuje pole, které je ekvivalentní poli JMS , je pole JMS mapováno na pole MQMD. Další pole MQMD jsou vystavena jako vlastnosti JMS , protože aplikace JMS může při komunikaci s jinou aplikací nežJMS potřebovat tato pole získat nebo nastavit.

Kopírování
V případě, že neexistuje žádný ekvivalent MQMD, je pole nebo vlastnost záhlaví JMS předána, případně transformována, jako pole uvnitř MQRFH2.

Záhlaví MQRFH2 a JMS
Tato kolekce témat popisuje záhlaví MQRFH verze 2, které obsahuje data specifická pro produkt JMS, jež jsou přidružena k obsahu zprávy. Záhlaví MQRFH verze 2 je rozšiřitelné a může také nést 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 formátování záhlaví 2.

Existují dvě části záhlaví, pevná část a proměnná část.

Pevná část
Pevná část je modelována podle vzoru záhlaví *standard* IBM MQ 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 definována.

Verze (MQLONG)
Číslo verze struktury.

Musí být MQRFH_VERSION_2 (hodnota: 2) (počáteční hodnota).

StrucLength (MQLONG)
Celková délka MQRFH2včetně datových polí NameValue.

Hodnota nastavená na StrucLength musí být násobkem 4 (data v polích NameValueData mohou být vyplněna mezerami).

Kódování (MQLONG)
Kódování dat.

Kódování libovolných číselných dat v části zprávy za položkou MQRFH2 (další záhlaví nebo data zprávy za tímto záhlavím).

CodedCharSetId (MQLONG)
Identifikátor kódované znakové sady.

Reprezentace libovolných znakových dat v části zprávy za položkou MQRFH2 (další záhlaví nebo data zprávy za tímto záhlavím).

Formát (MQCHAR8)
Název formátu.

Název formátu pro část zprávy následující za položkou MQRFH2.

Příznaky (MQLONG)
Příznaky.

MQRFH_NO_FLAGS = 0. Nejsou nastaveny žádné příznaky.

NameValueCCSID (MQLONG)
Identifikátor kódované znakové sady (CCSID) pro znakové řetězce NameValueData obsažené v tomto záhlaví. Data NameValuelze kódovat ve znakové sadě, která se liší od ostatních znakových řetězců obsažených v záhlaví (StrucID a formát).

Pokud je NameValueCCSID dvoubajtový Unicode CCSID (1200, 13488 nebo 17584), pořadí bajtů Unicode je stejné jako pořadí bajtů číselných polí v MQRFH2. (Například Version, StrucLengtha vlastní CCSID NameValue.)

Tabulka 16. Možné hodnoty pro pole NameValueCCSID

CCSID	**Význam**
1200	UTF-16, nejnovější podporovaná verze Unicode
13488	UTF-16, verze Unicode 2.0 dílčí sada

Tabulka 16. Možné hodnoty pro pole NameValueCCSID (pokračování)	
CCSID	**Význam**
17584	Podmnožina UTF-16, Unicode verze 3.0 (obsahuje symbol Euro)
1208	UTF-8, nejnovější podporovaná verze Unicode

Proměnná část

Proměnná část následuje za pevnou částí. Proměnná část obsahuje proměnný počet složek MQRFH2 . Každá složka obsahuje proměnný počet prvků nebo vlastností. Vlastnosti související se skupinou složek. Záhlaví MQRFH2 vytvořená produktem JMS může obsahovat libovolnou 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 17. `mcd` - název vlastnosti, synonymum, datový typ a složka			
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>`

Nepřidávejte své vlastní vlastnosti do složky `mcd`.

Složka `jms`

Produkt `jms` obsahuje pole záhlaví JMS a vlastnosti JMSX, které nelze plně vyjádřit v produktu `MQMD`. Složka `jms` je vždy přítomna v prostředí JMS `MQRFH2`.

Složka `usr`

`usr` obsahuje vlastnosti JMS definované aplikací přidružené ke zprávě. Složka `usr` je k dispozici pouze v případě, že aplikace nastavila vlastnost definovanou aplikací.

Složka `mqext`

Produkt `mqext` obsahuje následující typy vlastností:

- 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 18. mqext - název vlastnosti, synonymum, datový typ a složka			
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>

Nepřidávejte své vlastní vlastnosti do složky mqext.

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 19. mqps - název vlastnosti, synonymum, datový typ a složka			
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>

Nepřidávejte své vlastní vlastnosti do složky mqps.

Tabulka 20 na stránce 147 zobrazuje úplný seznam názvů vlastností.

Tabulka 20. Složky a vlastnosti MQRFH2 používané 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 20. Složky a vlastnosti MQRFH2 používané 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 (v)	i4
JMSCorrelationID	Řetězec	jms	CID	řetězec
JMSReplyTo	Místo určení	jms	Rto	řetězec
JMSTimestamp	long	jms	Tms (střední)	i8
JMSType.	Řetězec	mcd	Typ, nastavení, Fmt	řetězec
JMSXGroupID	Řetězec	jms	GID	řetězec
JMSXGroupSeq	celé číslo	jms	Pořadí	i4
xxx (definovaný uživatelem)	Libovolný	usr	xxx	jakékoli
		mcd	MSD	jms_none text jms_text bajtů jms_bytes ma_zpráv proud jms_stream objekt jms_object

NameValueDélka (MQLONG)
Délka datového řetězce NameValuev bajtech, která bezprostředně následuje za tímto polem délky (nezahrnuje svou vlastní délku).

NameValueData (MQCHARn)
Jeden znakový řetězec, jehož délka v bajtech je dána předchozím polem NameValueLength. Obsahuje složku obsahující posloupnost vlastností. Každá vlastnost je triplet name/type/value obsažený v prvku XML, jehož název je názvem složky, jak je uvedeno níže:

```
<foldername>
triplet1 triplet2 .....  tripletn </foldername>
```

Koncová značka `</foldername>` může být následována mezerami jako výplňovými znaky. 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 značkou `dt=` uveden jeden nebo více mezer.

name
je název vlastnosti; viz Tabulka 20 na stránce 147.

datatype
musí po přeskládání odpovídat jednomu z datových typů uvedených v části Tabulka 21 na stránce 149.

value
je řetězcová reprezentace hodnoty, která má být předána, pomocí definic v souboru Tabulka 21 na stránce 149.

Hodnota null je zakódována pomocí následující syntaxe:

```
<name dt='datatype' xsi:nil='true'></name>
```

Nepoužívejte soubor `xsi:nil='false'`.

Tabulka 21. Datové typy vlastností

Datový typ	Definice
řetězec	Libovolná posloupnost znaků kromě znaků < a &
typ boolean	Znak `0` nebo 1 (`0` = false, 1 = true)
bin.hex	Hexadecimální číslice představující oktety
i1	Číslo vyjádřené pomocí číslic `0..`9s volitelným znaménkem (bez zlomků nebo exponentu). Musí ležet v rozsahu -128 až 127 včetně
i2	Číslo vyjádřené pomocí číslic `0..`9s volitelným znaménkem (bez zlomků nebo exponentu). Musí ležet v rozsahu -32768 až 32767 včetně.
i4	Číslo vyjádřené pomocí číslic `0..`9s volitelným znaménkem (bez zlomků nebo exponentu). Musí ležet v rozsahu -2147483648 až 2147483647 včetně
i8	Číslo vyjádřené pomocí číslic `0..`9s volitelným znaménkem (bez zlomků nebo exponentu). Musí ležet v rozsahu -9223372036854775808 až 9223372036854775807 včetně
celé číslo	Číslo vyjádřené pomocí číslic `0..`9s volitelným znaménkem (bez zlomků nebo exponentu). Musí ležet ve stejném rozsahu jako i8. Tuto hodnotu lze použít namísto jednoho z typů i *, pokud odesílatel nechce k vlastnosti přidružit konkrétní přesnost.
r4	Číslo s pohyblivou řádovou čárkou, velikost < = 3.40282347E+38, > = 1.175E-37 vyjádřeno číslicemi `0..`9, volitelné znaménko, volitelné zlomkové číslice, volitelný exponent
r8	Číslo s pohyblivou řádovou čárkou, velikost < = 1.7976931348623E+308, > = 2.225E-307 vyjádřeno číslicemi `0..`9, volitelné znaménko, volitelné zlomkové číslice, volitelný exponent

Hodnota řetězce může obsahovat mezery. V řetězcové hodnotě musíte použít následující řídicí posloupnosti:

- `&` pro znak &
- `<` pro znak <

Můžete použít následující řídicí posloupnosti, ale nejsou požadovány:

- `>` pro znak >
- `&apos;` pro znak '
- `&quot;` pro znak "

JMS polí a vlastností s odpovídajícími poli MQMD
Tyto tabulky zobrazují pole MQMD odpovídající JMS polím záhlaví, JMS vlastnostem a JMS vlastnostem specifickým pro poskytovatele.

Tabulka 22 na stránce 150 vypíše pole záhlaví JMS a Tabulka 23 na stránce 150 vypíše vlastnosti JMS , které jsou mapovány přímo na pole MQMD. Tabulka 24 na stránce 150 uvádí vlastnosti specifické pro poskytovatele a pole MQMD, na která jsou mapovány.

Tabulka 22. Mapování polí záhlaví 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 23. Mapování vlastností 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 24. JMS mapování vlastností specifických pro poskytovatele 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_IBM_Report_Pass_Msg_ID	celé číslo	Sestava	MQLONG

Tabulka 24. JMS mapování vlastností specifických pro poskytovatele 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
Zpětná vazba JMS_IBM_Feedback	celé číslo	Zpětná vazba	MQLONG
Formát JMS_IBM_Format	Řetězec	Formát"1" na stránce 151	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 151	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 lze definovat nastavením vlastnosti JMS_IBM_Format pro zprávu (všimněte si, že existuje omezení na 8 znaků) nebo výchozí formát IBM MQ těla zprávy odpovídající typu zprávy JMS . Formát JMS_IBM_Format se mapuje na pole Formát MQMD pouze v případě, že zpráva neobsahuje žádné sekce RFH nebo RFH2 . V typické zprávě se mapuje na pole Formát na serveru RFH2 bezprostředně předcházejícím tělu zprávy.
2. Hodnota vlastnosti JMS_IBM_Character_Set je řetězcová hodnota, která obsahuje ekvivalent znakové sady Java pro číselnou hodnotu CodedCharacterSetId . Pole MQMD CodedCharacterSetId je číselná hodnota, která obsahuje ekvivalent řetězce znakové sady Java určeného 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 na pole MQMD a MQRFH2 v čase send () nebo publish ().

Tabulka 25 na stránce 152 ukazuje, jak jsou pole záhlaví JMS mapována na pole MQMD/RFH2 v čase send () nebo publish (). Tabulka 26 na stránce 152 ukazuje, jak jsou vlastnosti JMS mapovány do polí MQMD/RFH2 v čase send () nebo publish (). Tabulka 27 na stránce 153 ukazuje, jak jsou vlastnosti specifické pro poskytovatele JMS mapovány na pole MQMD v čase send () nebo publish (),

U polí označených Set by Message Object je přenesená hodnota hodnotou zadrženou ve zprávě JMS bezprostředně před operací send () nebo publish (). Hodnota ve zprávě JMS je operací ponechána beze změny.

Pro pole označená Nastavit metodou odeslání se hodnota přiřadí, když se provede odeslání () nebo publikování () (jakákoli hodnota ve zprávě JMS se ignoruje). Hodnota ve zprávě JMS se aktualizuje a zobrazí se použitá hodnota.

Pole označená jako pouze pro příjem se nepřenášejí a zůstávají beze změny ve zprávě pomocí send () nebo publish ().

Tabulka 25. Mapování polí odchozí zprávy

Název pole záhlaví JMS	Pole MQMD použité pro přenos	Header	Nastavil:
JMSDestination		MQRFH2	Metoda odeslání
JMSDeliveryMode	Trvání	MQRFH2	Metoda odeslání
JMSExpiration	Vypršení	MQRFH2	Metoda odeslání
JMSPriority.	Priorita	MQRFH2	Metoda odeslání
JMSMessageID	MsgID		Metoda odeslání
JMSTimestamp	PutDate/PutTime		Metoda odeslání
JMSCorrelationID	CorrelId	MQRFH2	Objekt zprávy
JMSReplyTo	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 určeného vlastností JMS_IBM_Character_Set.

Tabulka 26. Mapování vlastnosti JMS odchozí zprávy

JMS název vlastnosti	Pole MQMD použité pro přenos	Header	Nastavil:
JMSXUserID	UserIdentifier		Metoda odeslání
JMSXAppID	PutApplName		Metoda odeslání
JMSXDeliveryCount			Pouze příjem
JMSXGroupID	GroupId	MQRFH2	Objekt zprávy
JMSXGroupSeq	MsgSeqNumber	MQRFH2	Objekt zprávy

Poznámka:

Tyto vlastnosti jsou definovány jako jen pro čtení specifikací JMS a jsou nastaveny (v některých případech volitelně) poskytovatelem JMS.

V produktu IBM MQ classes for JMS lze dvě z těchto vlastností přepsat aplikací. Chcete-li tak učinit, ujistěte se, že místo určení bylo správně nakonfigurováno, a to nastavením následujících vlastností:

1. Nastavte vlastnost `WMQConstants.WMQ_MQMD_MESSAGE_CONTEXT` na hodnotu `WMQConstants.WMQ_MDCTX_SET_ALL_CONTEXT`.
2. Nastavte vlastnost `WMQConstants.WMQ_MQMD_WRITE_ENABLED` na hodnotu `true`.

Následující vlastnosti lze přepsat aplikací:

JMSXAppID
Tuto vlastnost lze přepsat nastavením vlastnosti `WMQConstants.JMS_IBM_MQMD_PUTAPPLNAME` ve zprávě-hodnota by měla být řetězec Java .

JMSXGroupID
Tuto vlastnost lze přepsat nastavením vlastnosti `WMQConstants.JMS_IBM_MQMD_GROUPID` ve zprávě-hodnota by měla být bajtové pole.

Tabulka 27. Mapování vlastností specifických pro poskytovatele JMS odchozí zprávy

Název vlastnosti specifické 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_IBM_Report_Pass_Msg_ID	Sestava		Objekt zprávy
JMS_IBM_Report_Pass_Correl_ID	Sestava		Objekt zprávy
JMS_IBM_Report_Discard_Msg	Sestava		Objekt zprávy
JMS_IBM_MsgType	MsgType		Objekt zprávy
Zpětná vazba JMS_IBM_Feedback	Zpětná vazba		Objekt zprávy
Formát JMS_IBM_Format	Formát		Objekt zprávy
Typ JMS_IBM_PutAppl	PutApplType		Metoda odeslání
JMS_IBM_Encoding	Kódování		Objekt zprávy
JMS_IBM_Character_Set	CodedCharacterSetId		Objekt zprávy
JMS_IBM_PutDate	PutDate		Metoda odeslání
JMS_IBM_PutTime	PutTime		Metoda odeslání
JMS_IBM_Last_Msg_In_Group	MsgFlags		Objekt zprávy

Mapování polí záhlaví JMS v send () nebo publish ()
Tyto poznámky se vztahují k mapování polí JMS na send () nebo publish ().

Místo určení JMSDestination pro MQRFH2
Tato hodnota je uložena jako řetězec, který serializuje charakteristické charakteristiky cílového objektu tak, aby přijímající objekt JMS mohl znovu vytvořit ekvivalentní cílový objekt. Pole MQRFH2 je kódováno jako identifikátor URI (podrobnosti o notaci identifikátoru URI viz "Identifikátory URI (Uniform Resource Identifier)" na stránce 215).

JMSReplyTo až MQMD.ReplyToQ, ReplyToQMgr, MQRFH2
Název fronty je zkopírován do produktu MQMD.ReplyToQ a název správce front se zkopírují do polí správce front ReplyTo. Informace o cílovém rozšíření (další užitečné podrobnosti, které jsou uchovány v cílovém objektu) se zkopírují do pole MQRFH2 . Pole MQRFH2 je kódováno jako identifikátor URI (podrobnosti o notaci identifikátoru URI viz "Identifikátory URI (Uniform Resource Identifier)" na stránce 215).

JMSDeliveryMode na MQMD.Persistence
Hodnota JMSDeliveryMode je nastavena pomocí metody send () nebo publish () nebo MessageProducer, pokud ji cílový objekt nepřepíše. Hodnota JMSDeliveryMode je mapována na MQMD.Persistence :

- Hodnota JMS PERSISTENT je ekvivalentní hodnotě MQPER_PERSISTENT.
- JMS hodnota NON_PERSISTENT je ekvivalentní hodnotě MQPER_NOT_PERSISTENT

Není-li vlastnost perzistence MQQueue nastavena na hodnotu WMQConstants.WMQ_PER_QDEF, je hodnota režimu doručení také zakódována v MQRFH2.

JMSExpiration do/z produktu MQMD.Expiry, MQRFH2
Funkce JMSExpiration ukládá dobu vypršení platnosti (součet aktuálního času a doby platnosti), zatímco funkce MQMD ukládá dobu platnosti. Také hodnota JMSExpiration je v milisekundách, ale MQMD.Expiry je v desetinách sekundy.

- Pokud metoda send () nastaví neomezenou dobu platnosti, MQMD.Expiry je nastavena na hodnotu MQEI_UNLIMITED a ve struktuře MQRFH2není zakódováno žádné vypršení platnosti JMSExpiration.
- Pokud metoda send () nastaví dobu platnosti, která je kratší než 214748364.7 sekund (přibližně 7 let), je doba platnosti uložena v produktu MQMD.Expirya doba vypršení platnosti (v milisekundách) je zakódována jako hodnota i8 v MQRFH2.
- Pokud metoda send () nastaví dobu platnosti na vyšší než 214748364.7 sekund, MQMD.Expiry je nastaveno na hodnotu MQEI_UNLIMITED. Skutečný čas vypršení platnosti v milisekundách je zakódován jako hodnota i8 v MQRFH2.

JMSPriority pro MQMD.Priority
Přímo mapujte hodnotu JMSPriority (0-9) na hodnotu priority MQMD (0-9). Je-li JMSPriority nastavena na jinou než výchozí hodnotu, je úroveň priority také zakódována v MQRFH2.

JMSMessageID z produktu 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 je předáno zpět aplikaci v poli JMSMessageID . IBM MQ messageId je 24bajtová binární hodnota, zatímco JMSMessageID je řetězec. JMSMessageID se skládá z binární hodnoty messageId převedené na posloupnost 48 hexadecimálních znaků s předponou ID znaků:. JMS poskytuje pokyn, který lze nastavit tak, aby zakázal produkci 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 ().

Pokud vyžadujete schopnost určit MQMD.MessageIDmůžete provést pomocí jednoho z rozšíření produktu IBM MQ JMS popsaných v části “Čtení a zápis deskriptoru zprávy z aplikace IBM MQ classes for JMS” na stránce 236.

JMSTimestamp na MQRFH2
Během odesílání je pole JMSTimestamp nastaveno podle hodin prostředí JVM. Tato hodnota je nastavena na MQRFH2. Jakákoli hodnota, která je nastavena do pole JMSTimestamp před přepsáním funkce send (). Viz také vlastnosti JMS_IBM_PutDate a JMS_IBM_PutTime .

JMSType na MQRFH2
Tento řetězec je nastaven do pole MQRFH2 mcd.Type . Pokud je ve formátu URI, může také ovlivnit pole mcd.Set a mcd.Fmt .

JMSCorrelationID pro MQMD.CorrelId, MQRFH2
JMSCorrelationID může obsahovat jednu z následujících položek:

ID zprávy specifické pro poskytovatele
Toto je identifikátor zprávy z dříve odeslané nebo přijaté zprávy, a proto by měl být řetězec 48 malých hexadecimálních číslic, které mají předponu `ID:` . Předpona je odebrána, zbývající znaky jsou převedeny na binární a pak jsou nastaveny na MQMD.CorrelId .

Nativní bajtová hodnota poskytovatele []
Hodnota se zkopíruje do MQMD.CorrelId je vyplněno nulami nebo v případě potřeby zkráceno na 24 bajtů. Ve struktuře MQRFH2není zakódována žádná hodnota CorrelId .

Řetězec specifický pro aplikaci
Hodnota se zkopíruje do MQRFH2. Prvních 24 bajtů řetězce ve formátu UTF8 je zapsáno do MQMD.CorrelID.

Mapování polí vlastností JMS
Tyto poznámky odkazují na mapování polí vlastností JMS ve zprávách IBM MQ .

JMSXUserID z MQMD UserIdentifier
JMSXUserID je nastaveno při návratu z volání send.

JMSXAppID z MQMD PutApplNázev
Hodnota JSMXAppID je nastavena při návratu z volání send.

JMSXGroupID do MQRFH2 (dvoubodové)
V případě zpráv typu point-to-point se hodnota JMSXGroupID zkopíruje do pole MQMD GroupID . Pokud JMSXGroupID začíná předponou ID:, převede se na binární. Jinak je zakódován jako řetězec UTF8 . Hodnota je vyplněna nebo zkrácena, je-li to nutné, na délku 24 bajtů. Příznak MQMF_MSG_IN_GROUP je nastaven.

JMSXGroupID pro MQRFH2 (publikování/odběr)
Pro zprávy publikování/odběru se JMSXGroupID zkopíruje do MQRFH2 jako řetězec.

JMSXGroupSeq MQMD MsgSeqPočet (dvoubodové)
Pro dvoubodové zprávy se hodnota JMSXGroupSeq zkopíruje do pole Počet MsgSeqdeskriptoru MQMD. Příznak MQMF_MSG_IN_GROUP je nastaven.

JMSXGroupSeq MQMD MsgSeqČíslo (publikování/odběr)
Pro zprávy publikování/odběru se JMSXGroupSeq zkopíruje do MQRFH2 jako i4.

Mapování JMS polí specifických pro poskytovatele
Následující poznámky odkazují na mapování polí specifických pro poskytovatele JMS do zpráv IBM MQ .

JMS_IBM_Report_*XXX* na sestavu MQMD
Aplikace JMS může nastavit volby sestavy MQMD pomocí následujících vlastností JMS_IBM_Report_*XXX* . Jediný modul MQMD je mapován na několik vlastností JMS_IBM_Report_*XXX* .

Konstanty JMS_IBM_Report_*XXX* jsou v `com.ibm.msg.client.jakarta.wmq.WMQConstants` nebo `com.ibm.msg.client.wmq.WMQConstants`.

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_IBM_Report_Pass_Msg_ID
MQRO_PASS_MSG_ID

JMS_IBM_Report_Pass_Correl_ID
MQRO_PASS_CORREL_ID

JMS_IBM_Report_Discard_Msg
MQRO_DISCARD_MSG

Hodnoty MQRO jsou v adresáři `com.ibm.mq.constants.CMQC`.

JMS_IBM_MsgType na MQMD MsgType
Hodnota se mapuje přímo na MQMD MsgType. Pokud aplikace nenastavila explicitní hodnotu JMS_IBM_MsgType, použije se výchozí hodnota. Tato výchozí hodnota je určena následujícím způsobem:

- Je-li parametr JMSReplyTo nastaven na cíl fronty IBM MQ , parametr MSGType je nastaven na hodnotu MQMT_REQUEST
- Není-li vlastnost JMSReplyTo nastavena nebo je-li nastavena na jinou hodnotu než na cíl fronty IBM MQ , je hodnota MsgType nastavena na hodnotu MQMT_DATAGRAM.

Zpětná vazba JMS_IBM_Feedback pro MQMD Feedback
Hodnota se mapuje přímo na MQMD Feedback.

Formát JMS_IBM_Format na formát MQMD
Hodnota se mapuje přímo na formát MQMD.

Kódování JMS_IBM_Encoding na MQMD
Je-li tato vlastnost nastavena, potlačí číselné kódování cílové fronty nebo tématu.

JMS_IBM_Character_Set na MQMD CodedCharacterSetId
Je-li tato vlastnost nastavena, potlačí 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 deskriptoru MQMD. Jakákoli hodnota, která je nastavena do vlastnosti JMS_IBM_PutDate před přepsáním odeslání. Toto pole je řetězec o délce osm znaků ve formátu data IBM MQ RRRRMMDD. Tuto vlastnost lze použít s vlastností JMS_IBM_PutTime k určení času vložení zprávy podle správce front.

JMS_IBM_PutTime z MQMD PutTime
Hodnota této vlastnosti je nastavena během odesílání přímo z pole PutTime v deskriptoru MQMD. Jakákoli hodnota, která je nastavena do vlastnosti JMS_IBM_PutTime před přepsáním odeslání. Toto pole je řetězec o délce osm znaků ve formátu času IBM MQ HHMMSSTH. Tuto vlastnost lze použít s vlastností JMS_IBM_PutDate k určení času vložení zprávy podle správce front.

JMS_IBM_Last_Msg_In_Group na 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 k označení starší aplikace IBM MQ , že tato zpráva je poslední ve skupině. Tato vlastnost je pro systém zpráv publikování/odběru ignorována.

Mapování polí IBM MQ na pole JMS (příchozí zprávy)
Tyto tabulky ukazují, jak jsou pole záhlaví a vlastnosti JMS mapována na pole MQMD a MQRFH2 v čase get () nebo receive ().

Tabulka 28 na stránce 156 ukazuje, jak jsou pole záhlaví JMS mapována na pole MQMD/MQRFH2 v čase get () nebo receive (). Tabulka 29 na stránce 157 ukazuje, jak jsou pole vlastností JMS mapována na pole MQMD/MQRFH2 v čase get () nebo receive (). Tabulka 30 na stránce 157 ukazuje, jak jsou mapovány vlastnosti specifické pro poskytovatele JMS .

Tabulka 28. Mapování pole záhlaví JMS příchozí zprávy

Název pole záhlaví JMS	Pole MQMD načteno z	Pole MQRFH2 načteno z
JMSDestination		jms.Dst nebo mqps.Top "1" na stránce 157
JMSDeliveryMode	Trvání"2" na stránce 157	jms.Dlv "2" na stránce 157

Tabulka 28. Mapování pole záhlaví JMS příchozí zprávy (pokračování)		
Název pole záhlaví JMS	**Pole MQMD načteno z**	**Pole MQRFH2 načteno z**
JMSExpiration		jms.Exp
JMSPriority.	Priorita	
JMSMessageID	MsgID	
JMSTimestamp	PutDate “2” na stránce 157 PutTime “2” na stránce 157	jms.Tms “2” na stránce 157
JMSCorrelationID	CorrelId “2” na stránce 157	jms.Cid “2” na stránce 157
JMSReplyTo	ReplyToQ “2” na stránce 157 ReplyToSprávce front “2” na stránce 157	jms.Rto “2” na stránce 157
JMSType.		mcd.Type, mcd.Set, mcd.Fmt
JMSRedelivered	BackoutCount	

Poznámka:

1. Pokud jsou nastaveny oba parametry jms.Dst a mqps.Top , použije se hodnota v poli jms.Dst .
2. Pro vlastnosti, které mohou mít hodnoty načtené z MQRFH2 nebo MQMD, je-li k dispozici obojí, použije se nastavení v MQRFH2 .
3. Hodnota vlastnosti JMS_IBM_Character_Set je řetězcová hodnota, která obsahuje ekvivalent znakové sady Java pro číselnou hodnotu CodedCharacterSetId .

Tabulka 29. Mapování vlastností příchozí zprávy		
JMS název vlastnosti	**Pole MQMD načteno z**	**Pole MQRFH2 načteno z**
JMSXUserID	UserIdentifier	
JMSXAppID	PutApplName	
JMSXDeliveryCount	BackoutCount	
JMSXGroupID	GroupId “1” na stránce 157	jms.Gid “1” na stránce 157
JMSXGroupSeq	MsgSeqPočet “1” na stránce 157	jms.Seq “1” na stránce 157

Poznámka:

1. Pro vlastnosti, které mohou mít hodnoty načtené z MQRFH2 nebo MQMD, je-li k dispozici obojí, použije se nastavení v MQRFH2 . Vlastnosti jsou nastaveny z hodnot MQMD pouze v případě, že jsou nastaveny příznaky zprávy MQMF_MSG_IN_GROUP nebo MQMF_LAST_MSG_IN_GROUP.

Tabulka 30. Mapování vlastností JMS specifických pro poskytovatele příchozích zpráv		
JMS název vlastnosti	**Pole MQMD načteno z**	**Pole MQRFH2 načteno z**
Výjimka JMS_IBM_Report_Exception	Sestava	
JMS_IBM_Report_Expiration	Sestava	
JMS_IBM_Report_COA	Sestava	

Tabulka 30. Mapování vlastností JMS specifických pro poskytovatele příchozích zpráv (pokračování)		
JMS název vlastnosti	**Pole MQMD načteno z**	**Pole MQRFH2 načteno z**
JMS_IBM_Report_COD	Sestava	
JMS_IBM_Report_PAN	Sestava	
JMS_IBM_Report_NAN	Sestava	
JMS_IBM_Report_ ID_Pass_Msg_ID	Sestava	
JMS_IBM_Report_Pass_Correl_ID	Sestava	
JMS_IBM_Report_Discard_Msg	Sestava	
JMS_IBM_MsgType	MsgType	
Zpětná vazba JMS_IBM_Feedback	Zpětná vazba	
Formát JMS_IBM_Format	Formát	
Typ JMS_IBM_PutAppl	PutApplType	
Kódování JMS_IBM_Encoding "1" na stránce 158	Kódování	
JMS_IBM_Character_Set "1" na stránce 158	CodedCharacterSetId	
JMS_IBM_PutDate	PutDate	
JMS_IBM_PutTime	PutTime	
JMS_IBM_Last_Msg_In_Group	MsgFlags	

1. Nastavte pouze v případě, že příchozí zpráva je bajtová zpráva.

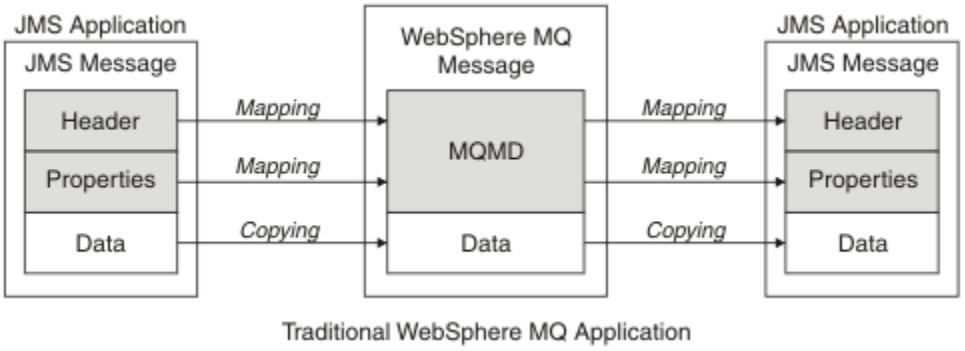
Výměna zpráv mezi aplikací JMS a tradiční aplikací IBM MQ
Toto téma popisuje, co se stane, když si aplikace JMS vymění zprávy s tradiční aplikací IBM MQ , která nemůže zpracovat záhlaví MQRFH2 .

Obrázek 11 na stránce 159 zobrazuje mapování.

Administrátor označuje, že aplikace JMS komunikuje s tradiční aplikací IBM MQ nastavením vlastnosti TARGCLIENT místa určení na hodnotu *MQ*. To znamená, že se nevytvoří žádné záhlaví MQRFH2 . Pokud se tak nestane, přijímající aplikace musí být schopna zpracovat záhlaví MQRFH2 .

Mapování z JMS na MQMD zaměřené na tradiční aplikaci IBM MQ je stejné jako mapování z JMS na MQMD zaměřené na aplikaci JMS . Pokud produkt IBM MQ classes for JMS obdrží zprávu IBM MQ s polem MQMD *Format* nastaveným na jinou hodnotu než MQFMT_RFH2, data jsou přijímána z jiné aplikace nežJMS . Pokud je formát MQFMT_STRING, zpráva se přijme jako textová zpráva JMS . Jinak je přijat jako JMS bajtová zpráva. Protože neexistuje žádný MQRFH2, lze obnovit pouze ty vlastnosti JMS , které jsou přenášeny v deskriptoru MQMD.

Pokud produkt IBM MQ classes for JMS obdrží zprávu, která nemá záhlaví MQRFH2 , vlastnost TARGCLIENT objektu Queue nebo Topic odvozenou z pole záhlaví JMSReplyTo zprávy je standardně nastavena na hodnotu MQ . To znamená, že zpráva odpovědi odeslaná do fronty nebo tématu také nemá záhlaví MQRFH2 . Toto chování zahrnutí záhlaví MQRFH2 do zprávy odpovědi můžete vypnout pouze v případě, že má původní zpráva záhlaví MQRFH2 , a to nastavením vlastnosti TARGCLIENTMATCHING továrny připojení na hodnotu NO.



Obrázek 11. Způsob transformace zpráv JMS na zprávy produktu IBM MQ bez záhlaví MQRFH2

Tělo zprávy JMS
Toto téma obsahuje informace o kódování samotného těla zprávy. Kódování závisí na typu zprávy JMS .

ObjectMessage
ObjectMessage je objekt serializovaný běhovým prostředím produktu Java běžným způsobem.

TextMessage
TextMessage je kódovaný řetězec. Pro odchozí zprávu je řetězec zakódován ve znakové sadě zadané cílovým objektem. Výchozí hodnota je UTF8 kódování (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. Takové znakové sady se používají hlavně při odesílání zprávy aplikaci, která neníJMS .

Pokud je znaková sada dvoubajtová (včetně UTF16), určuje pořadí bajtů specifikace kódování celého čísla cílového objektu.

Příchozí zpráva je interpretována pomocí znakové sady a kódování, které jsou uvedeny v samotné zprávě. Tyto specifikace jsou v posledním záhlaví IBM MQ (nebo MQMD, pokud nejsou žádná záhlaví). V případě zpráv systému JMS je posledním záhlavím obvykle záhlaví MQRFH2.

BytesMessage
BytesMessage je standardně posloupnost bajtů, jak je definována specifikací JMS 1.0.2 a přidruženou dokumentací Java .

Pro odchozí zprávu sestavenou samotnou aplikací lze použít vlastnost kódování cílového objektu k potlačení kódování celočíselných polí a polí s pohyblivou řádovou čárkou obsažených ve zprávě. Můžete například požadovat, aby byly hodnoty s pohyblivou řádovou čárkou uloženy ve formátu S/390 , nikoli ve formátu IEEE).

Příchozí zpráva je interpretována pomocí číselného kódování uvedeného v samotné zprávě. Tato specifikace je v posledním záhlaví IBM MQ (nebo MQMD, pokud neexistují žádná záhlaví). V případě zpráv systému JMS je posledním záhlavím obvykle záhlaví MQRFH2.

Pokud je přijata zpráva BytesMessage a je znovu odeslána bez úprav, její tělo je přeneseno bajtem pro bajt tak, jak bylo přijato. Vlastnost kódování cílového objektu nemá žádný vliv na tělo. Jedinou řetězcovou entitou, kterou lze explicitně odeslat v poli BytesMessage , je řetězec UTF8 . Tato hodnota je zakódována ve formátu Java UTF8 a začíná dvoubajtovým polem délky. Vlastnost znakové sady cílového objektu nemá žádný vliv na kódování odchozí zprávy BytesMessage. Hodnota znakové sady v příchozí zprávě IBM MQ nemá žádný vliv na interpretaci této zprávy jako JMS BytesMessage.

Aplikace jiné nežJava pravděpodobně nerozpoznají kódování Java UTF8 . Proto, aby aplikace JMS odeslala zprávu BytesMessage , která obsahuje textová data, musí sama převést své řetězce na bajtová pole a zapsat tato bajtová pole do BytesMessage.

MapMessage
MapMessage je řetězec obsahující triplety název/typ/hodnota XML kódovaný 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 seznamu Tabulka 21 na stránce 149. Výchozí datový typ je `string`, a proto je atribut `dt="string"` pro řetězcové prvky vynechán.

Znaková sada použitá k zakódování nebo interpretaci řetězce XML, který tvoří tělo zprávy mapy, je určena podle pravidel, která platí pro textovou zprávu.

Verze produktu IBM MQ classes for JMS starší než 5.3 zakódovali tělo zprávy mapy v následujícím formátu:

```
<map>
  <elementname1 dt="datatype1">value1</elementname1>
  <elementname2 dt="datatype2">value2</elementname2>
  ...
</map>
```

IBM MQ classes for JMS 5.3 a novější může interpretovat buď formát, ale verze produktu IBM MQ classes for JMS starší než 5.3 nemohou interpretovat aktuální formát.

Pokud aplikace potřebuje odeslat zprávy mapování do jiné aplikace, která používá verzi IBM MQ classes for JMS dřívější než 5.3, musí odesílající aplikace volat metodu továrny připojení `setMapNameStyle(WMQConstants.WMQ_MAP_NAME_STYLE_COMPATIBLE)` a určit, že zprávy mapování jsou odesílány v předchozím formátu. Standardně 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 seznamu Tabulka 21 na stránce 149. Výchozí datový typ je `string`, a proto je atribut `dt="string"` pro řetězcové prvky vynechán.

Znaková sada použitá k zakódování nebo interpretaci řetězce XML, který tvoří tělo StreamMessage , je určena podle pravidel, která platí pro TextMessage.

Pole MQRFH2.format je nastaveno takto:

MQFMT_NONE
pro ObjectMessage, BytesMessagenebo zprávy bez těla.

MQFMT_STRING
pro TextMessage, StreamMessagenebo MapMessage.

JMS převod zpráv
Převod dat zpráv v produktu JMS se provádí při odesílání a přijímání zpráv. Produkt IBM MQ provádí většinu převodu dat automaticky. Převádí textová a číselná data při přenosu zprávy mezi aplikacemi JMS . Text je převeden při výměně `JMSTextMessage` mezi aplikací JMS a aplikací IBM MQ .

Pokud plánujete provádět složitější výměny zpráv, zajímají vás následující témata. Komplexní výměny zpráv zahrnují:

- Přenos netextových zpráv mezi aplikací IBM MQ a aplikací JMS .
- Výměna textových dat v bajtovém formátu.
- Převod textu v aplikaci.

JMS Data zprávy

Převod dat je nezbytný pro výměnu textových a číselných dat mezi aplikacemi, a to i mezi dvěma aplikacemi JMS . Interní reprezentace textu a čísel musí být zakódována, aby je bylo možné přenést ve zprávě. Kódování vynutí rozhodnutí o tom, jak jsou reprezentována čísla a text. Produkt IBM MQ spravuje kódování textu a čísel ve zprávách systému JMS , s výjimkou `JMSObjectMessage`, viz “JMSObjectMessage” na stránce 167. Používá tři atributy zprávy. Tyto tři atributy jsou `CodedCharacterSetId`, `Encoding`a `Format`.

Tyto tři atributy zprávy jsou obvykle uloženy v polích JMS záhlaví `MQRFH2`zprávy JMS . Pokud je typ zprávy `MQ`spíše než JMS typ zprávy, atributy jsou uloženy v deskriptoru zprávy `MQMD`. Atributy se používají k převodu dat zprávy JMS . Data zprávy JMS se přenášejí v datové části zprávy IBM MQ .

JMS Vlastnosti zprávy

Vlastnosti zprávy JMS , jako např. `JMS_IBM_CHARACTER_SET`, se vyměňují v části záhlaví `MQRFH2` zprávy JMS , pokud nebyla zpráva odeslána bez `MQRFH2`. Pouze `JMSTextMessage` a `JMSBytesMessage` mohou být odeslány bez `MQRFH2`. Pokud je vlastnost JMS uložena jako vlastnost zprávy IBM MQ v deskriptoru zprávy `MQMD`, je převedena jako součást převodu `MQMD` . Pokud je vlastnost JMS uložena v souboru `MQRFH2`, je uložena ve znakové sadě určené parametrem `MQRFH2.NameValueCCSID`. Když je zpráva odeslána nebo přijata, vlastnosti zprávy jsou převedeny na a z jejich interního znázornění v prostředí JVM. Převod je do a ze znakové sady deskriptoru zprávy nebo `MQRFH2.NameValueCCSID`. Číselná data jsou převedena na text.

JMS převod zpráv

Následující témata obsahují příklady a úlohy, které jsou užitečné, pokud plánujete výměnu složitějších zpráv, které vyžadují převod.

JMS přístupů k převodu zpráv
Návrhářům aplikací JMS je otevřena řada přístupů pro převod dat. Tyto přístupy nejsou výlučné; některé aplikace pravděpodobně používají kombinaci těchto přístupů. Pokud vaše aplikace pouze mění text nebo si vyměňují zprávy pouze s jinými aplikacemi JMS , obvykle převod dat nepovažujete za možný. Převod dat se provádí automaticky za vás, pomocí IBM MQ.

Můžete se zeptat na několik otázek, jak přistupovat k převodu zpráv:

Je třeba vůbec přemýšlet o konverzi zpráv?
V některých případech, například při přenosu zpráv z JMS do JMS , a při výměně textových zpráv s programy IBM MQ provádí produkt IBM MQ automaticky nezbytné převody. Možná budete chtít řídit převod dat z výkonnostních důvodů, nebo si budete vyměňovat složité zprávy, které mají předdefinovaný formát. V případech, jako jsou tyto, musíte porozumět převodu zpráv a přečíst si následující témata.

Jaké druhy obrácení existují?
Existují čtyři hlavní typy konverze, které jsou vysvětleny v následujících sekcích:

1. “Převod dat klienta JMS” na stránce 162
2. “Převod dat aplikace” na stránce 162
3. “Převod dat správce front” na stránce 163
4. “Převod dat kanálu zpráv” na stránce 163

Kde se má provést konverze?
Část “Výběr přístupu k převodu zpráv: přijímač je dobrý” na stránce 164popisuje obvyklý přístup "příjemce je dobrý". "Přijímač je dobrý" platí také pro převod dat JMS .

Převod dat klienta JMS

JMS klient[1]převod dat je převod primitiv Java a objektů na bajty ve zprávě systému JMS , jak jsou odeslány na místo určení, a převod zpět, když je přijat. Převod dat klienta JMS používá metody tříd JMSMessage . Metody jsou uvedeny podle typu třídy JMSMessage v souboru Tabulka 31 na stránce 165.

Převod na a z interní reprezentace čísel a textu prostředí JVM se provádí pro metody čtení, získání, nastavení a zápisu. Konverze se provádí při odeslání zprávy a při volání některé z metod read nebo get pro přijatou zprávu.

Kódová stránka a číselné kódování použité k zápisu nebo nastavení obsahu zprávy jsou definovány jako atributy místa určení. Kódovou stránku místa určení a číselné kódování lze administrativně změnit. Aplikace může také přepsat cílovou kódovou stránku a kódování nastavením vlastností zprávy, které řídí zápis nebo nastavení obsahu zprávy.

Chcete-li při odesílání zprávy JMSBytesMessage do místa určení, které není definováno jako kódování Native , převést kódování čísel, musíte před odesláním zprávy nastavit vlastnost zprávy JMS_IBM_ENCODING . Pokud sledujete program "receiver makes good" pattern, nebo pokud si vyměňujete zprávy mezi aplikacemi JMS , aplikace nemusí nastavit JMS_IBM_ENCODING. Ve většině případů můžete vlastnost Encoding ponechat jako Native.

Pro zprávy JMSStreamMessage, JMSMapMessagea JMSTextMessage jsou použity 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 nastavit hodnotu JMS_IBM_CHARACTER_SET , má-li se použít vlastnost cílové znakové sady.

Chcete-li získat data ve zprávě, aplikace volá metody read nebo get zprávy JMS . Metody odkazují na kódovou stránku a kódování definované v předchozím záhlaví zprávy, aby správně vytvořily primitiva a objekty Java .

Převod dat klienta JMS splňuje potřeby většiny aplikací JMS , které si vyměňují zprávy mezi klientem JMS . Nekódujte žádný explicitní převod dat. Nepoužíváte třídu java.nio.charset.Charset , která se obvykle používá při zápisu textu do souboru. Metody writeString a setString za vás provádí převod.

Další podrobnosti o převodu dat klienta JMS viz “JMS převod a kódování zpráv klienta” na stránce 173.

Převod dat aplikace

Klientská aplikace JMS může provádět explicitní převod znakových dat pomocí třídy java.nio.charset.Charset ; viz příklady v Obrázek 14 na stránce 166 a Obrázek 15 na stránce 166. Řetězcová data se převedou na bajty pomocí metody getBytes a odešlou se jako bajty. Bajty jsou převedeny zpět na text pomocí konstruktoru String , který přebírá bajtové pole a Charset. Znaková data se převedou 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]. Můžete také odesílat a přijímat bajty pomocí JMSStreamMessage, JMSMapMessagenebo JMSObjectMessage.

Neexistují žádné metody Java pro kódování a dekódování bajtů, které obsahují číselná data reprezentovaná v různých formátech kódování. Číselná data jsou zakódována a dekódována automaticky pomocí číselných JMSMessage metod čtení a zápisu. Metody čtení a zápisu používají hodnotu atributu JMS_IBM_ENCODING dat zprávy.

Typické použití pro převod dat aplikace je v případě, že klient JMS odešle nebo přijme formátovanou zprávu od jiné aplikace nežJMS . Formátovaná zpráva obsahuje textová, číselná a bajtová data uspořádaná podle délky datových polí. Pokud aplikace, která není typuJMS , neurčila formát zprávy jako "MQSTR ", je zpráva sestavena jako JMSBytesMessage. Chcete-li přijímat formátovaná data zprávy v systému JMSBytesMessage , musíte volat posloupnost metod. Metody musí být volány ve stejném pořadí, v jakém byla pole zapsána do zprávy. Pokud jsou pole číselná, musíte znát kódování a délku číselných

[1] "JMS Klient" odkazuje na IBM MQ classes for JMS , které implementují rozhraní JMS , které je spuštěno buď v režimu klienta, nebo v režimu vazeb.

[2] Jedna výjimka: Data zapsaná pomocí writeUTF začínají dvoubajtovým polem délky

dat. Pokud jakékoli pole obsahuje bajtová nebo textová data, musíte znát délku všech bajtových dat ve zprávě. Existují dva způsoby, jak převést formátovanou zprávu na objekt Java , který se snadno používá.

1. Vytvořte třídu Java odpovídající záznamu, abyste zapouzdřili čtení a zapsání zprávy. Přístup k datům v záznamu je pomocí metod get a set třídy.
2. Vytvořte třídu Java odpovídající záznamu rozšířením třídy `com.ibm.mq.headers` . Přístup k datům ve třídě je s přistupujícími objekty typu formuláře, `getStringValue(`*`fieldname`*`);`

Viz “Výměna formátovaného záznamu s jinou aplikací nežJMS” na stránce 181.

Převod dat správce front

Převod kódové stránky může být proveden správcem front, když klientský program JMS obdrží zprávu. Konverze je stejná jako konverze provedená pro program v jazyce C. Program v jazyce C nastaví `MQGMO_CONVERT` jako volbu parametru `MQGET` `GetMsgOpts` ; viz Obrázek 13 na stránce 166. Správce front provádí převod pro klientský program systému JMS , který přijímá zprávu, pokud je vlastnost cíle `WMQ_RECEIVE_CONVERSION` nastavena na hodnotu `WMQ_RECEIVE_CONVERSION_QMGR`, klientský program JMS může také nastavit vlastnost cíle; viz Obrázek 12 na stránce 163.

```
((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 výhoda převodu správce front přichází při výměně zpráv s jinými aplikacemi nežJMS . Pokud je ve zprávě definováno pole `Format` a cílová znaková sada nebo kódování se liší od zprávy, provede správce front převod dat pro cílovou aplikaci, pokud to aplikace požaduje. Správce front převádí data zpráv formátovaná podle jednoho z předdefinovaných typů zpráv IBM MQ , například záhlaví CICS bridge (`MQCIH`). Je-li pole `Format` definováno uživatelem, správce front vyhledá uživatelskou proceduru pro převod dat s názvem uvedeným v poli `Format` .

Převod dat správce front se používá pro nejlepší efekt s návrhovém vzorem "příjemce je dobrý" . Odesílající klient JMS nemusí provádět převod. Přijímající program, který neníJMS , spoléhá na uživatelskou proceduru převodu, aby se ujistil, že zpráva je doručena v požadované kódové stránce a kódování. U odesílajícího klienta JMS a příjemce, který neníJMS , se příklad použije na IBM MQ.

Pomocí obslužného programu uživatelské procedury pro převod dat **`crtmqcvx`**můžete vytvořit uživatelskou proceduru pro převod dat, která umožní správci front převést vlastní data ve formátu záznamu. Můžete sestavit vlastní formát záznamu, použít `com.ibm.mq.headers` pro přístup jako třídu Java a použít vlastní uživatelskou proceduru převodu. V systému z/OS se obslužný program nazývá **`CSQUCVX`**a v systému IBM i **`CVTMQMDTA`**. Viz “Výměna formátovaného záznamu s jinou aplikací nežJMS” na stránce 181.

Převod dat kanálu zpráv

IBM MQ Kanály odesilatele, serveru, příjemce klastru a odesilatele klastru mají volbu převodu zpráv `CONVERT`. Obsah zprávy lze volitelně převést při odeslání zprávy. Převod probíhá na odesílajícím konci kanálu. Definice příjemce klastru se používá k automatickému definování odpovídajícího odesílacího kanálu klastru.

Konverze dat kanály zpráv se obvykle používá, pokud není možné použít jiné formy konverze.

Výběr přístupu k převodu zpráv: "přijímač je dobrý"

Obvyklý přístup v návrhu aplikace IBM MQ pro převod kódu je "přijímač je dobrý". "Přijímač dělá dobře" snižuje počet konverzí zpráv. Také se vyhýbá problému s neočekávanými chybami kanálu v případě, že dojde k selhání převodu zpráv v některém zprostředkovatelském správci front během přenosu zpráv. Pravidlo "receiver make good" je porušeno pouze v případě, že existuje důvod, proč příjemce nemůže provést dobro. Přijímající platforma nemusí mít například správnou znakovou sadu.

"Přijímač je dobrý" je také dobrým obecným vodítkem pro klientské aplikace JMS . Ve specifických případech však může být efektivnější převod na správnou znakovou sadu u zdroje. Při odesílání zprávy obsahující text nebo číselné typy musí dojít k převodu z interní reprezentace prostředí JVM. Převod na znakovou sadu požadovanou příjemcem, pokud příjemce není klientem JMS , může odstranit potřebu, aby příjemce, který není příjemcem systémuJMS , provedl převod. Pokud je příjemcem klient JMS , bude se přesto znovu převádět, aby se dekódovala data zprávy a vytvořily se primitiva a objekty Java .

Rozdíl mezi klientskými aplikacemi JMS a aplikacemi napsanými v jazyce, jako je C, je v tom, že produkt Java musí provádět převod dat. Aplikace v jazyce Java musí převádět čísla a text z interní reprezentace do kódovaného formátu používaného ve zprávách.

Nastavením místa určení nebo vlastností zprávy můžete nastavit znakovou sadu a kódování používané produktem IBM MQ k kódování čísel a textu ve zprávách. Za normálních okolností byste nechali znakovou sadu jako `1208` a kódování jako `Native`.

IBM MQ nepřevádí bajtová pole. Chcete-li kódovat řetězce a znaková pole do bajtových polí, použijte balík `java.nio.charset` . `Charset` uvádí znakovou sadu použitou k převodu řetězce nebo znakového pole na bajtové pole. Bajtové pole můžete také dekódovat na řetězec nebo znakové pole pomocí `Charset`. Při kódování řetězců a znakových polí není dobrým zvykem spoléhat se na `java.nio.charset.Charset.defaultCodePage` . Výchozí hodnota `Charset` je obvykle `windows-1252` on Windowsa `UTF-8` on AIX and Linux. `windows-1252` je jednobajtová znaková sada a `UTF-8` je vícebajtová znaková sada.

Při výměně zpráv s jinými aplikacemi JMS obvykle ponechte vlastnosti cílové znakové sady a kódování na výchozích hodnotách `UTF-8` a `Native` . Pokud si s aplikací JMS vyměňujete zprávy obsahující čísla nebo text, vyberte jeden z typů zpráv `JMSTextMessage`, `JMSStreamMessage`, `JMSMapMessage`nebo `JMSObjectMessage` , které odpovídají vašemu účelu. Nejsou k dispozici žádné další úlohy převodu.

Pokud si vyměňujete zprávy s jinými aplikacemi nežJMS , které používají formát záznamu, je to složitější. Pokud celý záznam neobsahuje text a může být přenesen jako `JMSTextMessage`, musíte zakódovat a dekódovat text v aplikaci. Nastavte typ cílové zprávy na `MQ`a použijte `JMSBytesMessage` , abyste se vyvarovali IBM MQ classes for JMS přidání dalších informací o záhlaví a značkách do dat zprávy. K zápisu čísel a bajtů použijte metody `JMSBytesMessage` a třída `Charset` explicitně převede text na bajtová pole. Výběr znakové sady může ovlivnit několik faktorů:

- Výkon: Můžete snížit počet konverzí transformací textu na znakovou sadu, která se používá na největším počtu serverů?
- Uniformita: Přenese všechny zprávy ve stejné znakové sadě.
- Bohatost: Jaké znakové sady mají všechny kódové body, které musí aplikace používat?
- Jednoduchost: Jednobajtové znakové sady se používají jednodušeji než proměnné délky a vícebajtové znakové sady.

Viz “Výměna formátovaného záznamu s jinou aplikací nežJMS” na stránce 181. pro příklady převodu zpráv vyměňovaných s jinými aplikacemi nežJMS .

Příklady

Tabulka typů zpráv a typů převodů

Tabulka 31. Typy zpráv a typy převodů

	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

```
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 souboru JMSBytesMessage

Kód v poli Obrázek 14 na stránce 166 odešle řetězec v poli BytesMessage. Pro jednoduchost příklad odešle jeden řetězec, pro který je vhodnější `JMSTextMessage` . Chcete-li přijmout textový řetězec v bajtech obsahující směs typů, musíte znát délku řetězce v bajtech s názvem *TEXT_LENGTH* in Obrázek 15 na stránce 166. I pro řetězec s pevným počtem znaků může být délka bajtové reprezentace 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 souboru `String` v adresáři `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

JMS převod a kódování zpráv klienta
Jsou uvedeny metody, které používáte k provedení převodu a kódování zpráv klienta JMS , s příklady kódu pro každý typ převodu.

Převod dat správce front
Převod dat správce front byl vždy k dispozici pro jiné aplikace nežJMS , které přijímají zprávy od klientů systému JMS . Klienti systému JMS , kteří přijímají zprávy, také používají převod dat správce front, který je volitelný.

Související úlohy

Výměna formátovaného záznamu s jinou aplikací nežJMS
Postupujte podle kroků navržených v této úloze, chcete-li navrhnout a sestavit uživatelskou proceduru pro převod dat a klientskou aplikaci JMS , která si může vyměňovat zprávy s jinou aplikací nežJMS pomocí produktu JMSBytesMessage. K výměně formátované zprávy s jinou aplikací nežJMS může dojít s voláním uživatelské procedury pro převod dat nebo bez volání.

Související odkazy

JMS typy zpráv a převod
Volba typu zprávy ovlivňuje váš přístup k převodu zpráv. Interakce převodu zpráv a typu zpráv je popsána pro JMS typy zpráv JMSObjectMessage, JMSTextMessage, JMSMapMessage, JMSStreamMessagea JMSBytesMessage.

JMS typy zpráv a převod
Volba typu zprávy ovlivňuje váš přístup k převodu zpráv. Interakce převodu zpráv a typu zpráv je popsána pro JMS typy zpráv JMSObjectMessage, JMSTextMessage, JMSMapMessage, JMSStreamMessagea JMSBytesMessage.

JMSObjectMessage

JMSObjectMessage obsahuje jeden objekt a všechny objekty, na které odkazuje, jsou prostředím JVM serializovány do proudu bajtů. Text je serializován do adresáře UTF-8a je omezen na řetězce nebo znaková pole o velikosti nejvýše 65534 bajtů. Výhodou produktu JMSObjectMessage je, že aplikace nejsou zapojeny do žádných problémů s převodem dat, pokud používají pouze metody a atributy objektu. Produkt JMSObjectMessage poskytuje převod dat pro komplexní objekty bez aplikačního programátora s ohledem na to, jak kódovat objekt ve zprávě. Nevýhodou použití produktu JMSObjectMessage je, že jej lze vyměnit pouze s jinými aplikacemi JMS . Výběrem jednoho z ostatních typů zpráv JMS je možné vyměňovat JMS zprávy s jinými aplikacemi nežJMS .

"Odeslání a přijetí JMSObjectMessage" na stránce 169 ukazuje objekt String , který je vyměňován ve zprávě.

Aplikace klienta JMS může přijmout JMSObjectMessage pouze ve zprávě, která má tělo ve stylu JMS. Cíl musí určovat tělo stylu JMS .

JMSTextMessage

JMSTextMessage obsahuje jeden textový řetězec. Když je odeslána textová zpráva, text Format je nastaven na "MQSTR ", WMQConstants.MQFMT_STRING. Parametr CodedCharacterSetId textu je nastaven na identifikátor kódované znakové sady definovaný pro místo určení. Text je zakódován do souboru CodedCharacterSetId pomocí IBM MQ. Pole CodedCharacterSetId a Format jsou buď nastavena v deskriptoru zprávy, MQMD, nebo do polí JMS v MQRFH2. Pokud je zpráva definována tak, že má styl těla zprávy WMQ_MESSAGE_BODY_MQ , nebo není specifikován styl těla zprávy, ale cílové místo určení je WMQ_TARGET_DEST_MQ, pak jsou nastavena pole deskriptoru zprávy. Jinak má zpráva JMS RFH2 a pole jsou nastavena v pevné části MQRFH2.

Aplikace může přepsat identifikátor kódované znakové sady definovaný pro místo určení. Musí nastavit vlastnost zprávy JMS_IBM_CHARACTER_SET na identifikátor kódované znakové sady; viz příklad v části "Odeslání a přijetí JMSTextmessage" na stránce 170.

Když klient JMS volá převod správce front metody consumer.receive , je volitelný. Převod správce front je povolen nastavením vlastnosti místa určení WMQ_RECEIVE_CONVERSION na hodnotu WMQ_RECEIVE_CONVERSION_QMGR. Správce front před přenosem zprávy do klienta JMS převádí textovou zprávu ze souboru JMS_IBM_CHARACTER_SET určeného pro danou zprávu. Znaková sada převedené zprávy je 1208, UTF-8, pokud cíl nemá jiný WMQ_RECEIVE_CCSID. Hodnota CodedCharacterSetId ve zprávě, která odkazuje na JMSTextMessage , je aktualizována na ID cílové znakové sady. Text je dekódován z cílové znakové sady do Unicode metodou getText ; viz příklad v "Odeslání a přijetí JMSTextmessage" na stránce 170.

JMSTextMessage lze odeslat v těle zprávy ve stylu MQbez záhlaví JMS MQRFH2 .
Hodnota atributů místa určení WMQ_MESSAGE_BODY a WMQ_TARGET_DEST určuje styl těla

zprávy, není-li přepsán aplikací. Aplikace může přepsat hodnoty nastavené v cíli voláním funkce `destination.setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_MQ)` nebo `destination.setTargetClient(WMQConstants.WMQ_TARGET_DEST_MQ)`.

Odešlete-li objekt `JMSTextMessage` s tělem stylu MQ odesláním do místa určení s volbou `WMQ_MESSAGE_BODY` nastavenou na hodnotu `WMQ_MESSAGE_BODY_MQ`, nelze jej přijmout jako objekt `JMSTextMessage` ze stejného místa určení. Všechny zprávy přijaté z místa určení s hodnotou `WMQ_MESSAGE_BODY` nastavenou na hodnotu `WMQ_MESSAGE_BODY_MQ` jsou přijímány jako `JMSBytesMessage`. Pokud se pokusíte přijmout zprávu jako `JMSTextMessage` , způsobí výjimku `ClassCastException: com.ibm.jms.JMSBytesMessage cannot be cast to jakarta (or javax).jms.TextMessage`.

Poznámka: Text v souboru `JMSBytesMessage` není převeden klientem JMS. Klient může přijmout text ve zprávě pouze jako bajtové pole. Je-li povolen převod správce front, je text převeden správcem front, ale klient JMS jej musí přesto přijmout jako bajtové pole v souboru `JMSBytesMessage`.

Obecně je lepší použít vlastnost `WMQ_TARGET_DEST` k řízení toho, zda se má `JMSTextMessage` odeslat se stylem těla produktu MQ nebo JMS. Poté můžete přijmout zprávu z místa určení, které má hodnotu `WMQ_TARGET_DEST` nastavenou na `WMQ_TARGET_DEST_MQ` nebo `WMQ_TARGET_DEST_JMS`. `WMQ_TARGET_DEST` nemá na příjemce žádný vliv.

JMSMapMessage a JMSStreamMessage

Tyto dva typy zpráv JMS jsou podobné. Do zpráv můžete číst a zapisovat primitivní typy pomocí metod založených na rozhraních `DataInputStream` a `DataOutputStream` ; viz “Tabulka typů zpráv a typů převodů” na stránce 172. Podrobnosti jsou popsány v části “JMS převod a kódování zpráv klienta” na stránce 173. Každé primitivum je označeno; viz “Tělo zprávy JMS” na stránce 159.

Číselná data se čtou a zapisují do zprávy kódované jako text XML. Na vlastnost místa určení `JMS_IBM_ENCODING`se neodkazuje. S textovými daty se zachází stejně jako s textem v souboru `JMSTextMessage`. Pokud byste se podívali na obsah zprávy vytvořený v příkladu v souboru Obrázek 20 na stránce 170, všechna data zprávy by byla v EBCDIC, protože byla odeslána s hodnotou znakové sady 37.

Můžete odeslat více položek v `JMSMapMessage` nebo `JMSStreamMessage`.

Jednotlivé datové položky můžete načíst podle názvu z `JMSMapMessage`nebo podle pozice z `JMSStreamMessage`. Každá položka je dekódována při volání metody get nebo read pomocí hodnoty `CodedCharacterSetId` uložené ve zprávě. Pokud metoda použitá k načtení položky vrátí jiný typ než typ, který byl odeslán, typ se převede. Pokud typ nelze převést, dojde k výjimce. Podrobnosti viz Třída JMSStreamMessage . Příklad v souboru “Odeslání dat v JMSStreamMessage a JMSMapMessage” na stránce 170 ilustruje převod typu a získání obsahu `JMSMapMessage` mimo pořadí.

Pole `MQRFH2.format` pro `JMSMapMessage` a `JMSStreamMessage` je nastaveno na "`MQSTR  `". Je-li vlastnost místa určení `WMQ_RECEIVE_CONVERSION` nastavena na hodnotu `WMQ_RECEIVE_CONVERSION_QMGR`, správce front data zprávy před odesláním klientovi JMS převede. `MQRFH2.CodedCharacterSetId` zprávy je `WMQ_RECEIVE_CCSID` místa určení. Parametr `MQRFH2.Encoding` má hodnotu `Native`. Je-li hodnota `WMQ_RECEIVE_CONVERSION` `WMQ_RECEIVE_CONVERSION_CLIENT_MSG` `CodedCharacterSetId` a `Encoding` parametru `MQRFH2` je hodnota nastavená odesilatelem.

Aplikace klienta JMS může přijmout `JMSMapMessage` nebo `JMSStreamMessage` pouze ve zprávě, která má tělo ve stylu JMS, a z místa určení, které neurčuje tělo stylu MQ .

JMSBytesMessage

`JMSBytesMessage` může obsahovat více primitivních typů. Do zpráv můžete číst a zapisovat primitivní typy pomocí metod založených na rozhraních `DataInputStream` a `DataOutputStream` ; viz “Tabulka typů zpráv a typů převodů” na stránce 172. Podrobnosti jsou popsány v části “JMS typy zpráv a převod” na stránce 167.

Kódování číselných dat ve zprávě je řízeno hodnotou JMS_IBM_ENCODING , která je nastavena před zápisem číselných dat do souboru JMSBytesMessage. Aplikace může přepsat výchozí Native kódování definované pro JMSBytesMessage nastavením vlastnosti zprávy JMS_IBM_ENCODING.

Textová data lze číst a zapisovat v produktu UTF-8 pomocí metod readUTF a writeUTFnebo v kódování Unicode pomocí metod readChar a writeChar . Neexistují žádné metody, které by používaly CodedCharacterSetId. Volitelně může klient JMS zakódovat a dekódovat text do bajtů pomocí třídy Charset . Přenáší bajty mezi prostředím JVM a zprávou bez toho, aby IBM MQ classes for JMS provádělo jakýkoli převod; viz “Odesílání a příjem textu v souboru JMSBytesMessage” na stránce 170.

Soubor JMSBytesMessage odeslaný do aplikace MQ je obvykle odeslán v těle zprávy ve stylu MQbez záhlaví JMS MQRFH2 . Pokud je odeslán do aplikace JMS , styl těla zprávy je obvykle JMS. Hodnota atributů místa určení WMQ_MESSAGE_BODY a WMQ_TARGET_DEST určuje styl těla zprávy, není-li přepsán aplikací. Aplikace může přepsat hodnoty nastavené v cíli voláním funkce destination.setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_MQ) nebo destination.setTargetClient(WMQConstants.WMQ_TARGET_DEST_MQ).

Pokud odešlete zprávu JMSBytesMessage s tělem stylu MQ , můžete obdržet zprávu z místa určení, které definuje buď styl těla zprávy MQ , nebo styl těla zprávy JMS . Pokud odešlete zprávu JMSBytesMessage s tělem stylu JMS, musíte obdržet zprávu z místa určení, které definuje styl těla zprávy JMS . Pokud tak neučiníte, bude s produktem MQRFH2 zacházeno jako s částí dat uživatelské zprávy, což nemusí být to, co očekáváte.

Bez ohledu na to, zda má zpráva styl těla MQ nebo JMS , způsob, jakým je přijata, není ovlivněn nastavením WMQ_TARGET_DEST.

Zpráva může být později transformována správcem front, pokud je pro data zprávy dodána položka Format a je povolen převod dat správce front. Pole formátu nepoužívejte pro nic jiného než pro určení formátu dat zprávy, nebo jej ponechte prázdné, MQConstants.MQFMT_NONE

V produktu JMSBytesMessagemůžete odeslat více položek. Každá číselná položka je převedena, když je zpráva odeslána pomocí kódování definovaného pro zprávu.

Jednotlivé položky dat můžete načíst z JMSBytesMessage. Volat metody čtení ve stejném pořadí, v jakém byly metody zápisu volány pro vytvoření zprávy. Každá číselná položka je převedena, když je zpráva volána pomocí hodnoty Encoding uložené ve zprávě.

Na rozdíl od JMSMapMessage a JMSStreamMessageobsahuje JMSBytesMessage pouze data zapsaná aplikací. Do dat zprávy nejsou uložena žádná další data, jako např. značky XML použité k definování položek v JMSMapMessage a JMSStreamMessage. Z tohoto důvodu použijte produkt JMSBytesMessage k přenosu zpráv formátovaných pro jiné aplikace.

Převod mezi JMSBytesMessage a DataInputStream a DataOutputStream je v některých aplikacích užitečný. Kód založený na příkladu “Čtení a zápis zpráv pomocí DataInputStream a DataOutputStream” na stránce 171je nezbytný pro použití balíku com.ibm.mq.header s JMS.

Příklady

Odeslání a přijetí 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. Odeslání a přijetí JMSObjectMessage

Odeslání a přijetí JMSTextmessage

Textová zpráva nemůže obsahovat text v různých znakových sadách. Příklad zobrazuje text v různých znakových sadách, odesílaný 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 cílem

```
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 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

Odeslání 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 adresáři JMSStreamMessage a JMSMapMessage

Odesílání a příjem textu v souboru JMSBytesMessage

Kód v poli Obrázek 21 na stránce 171 odešle řetězec v poli BytesMessage. Pro jednoduchost příklad odešle jeden řetězec, pro který je vhodnější JMSTextMessage . Chcete-li přijmout textový řetězec

v bajtech obsahující směs typů, musíte znát délku řetězce v bajtech s názvem *TEXT_LENGTH* in Obrázek 22 na stránce 171. I pro řetězec s pevným počtem znaků může být délka bajtové reprezentace 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 souboru `String` v adresáři `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 souboru Obrázek 23 na stránce 171 vytvoří 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 nastavuje vlastnost JMS_IBM_ENCODING , je označen jako komentář. Příkaz je platný, pokud zapisujete přímo do JMSBytesMessage, ale nemá žádný vliv na zápis do DataOutputStream. Čísla zapsaná do souboru DataOutputStream jsou zakódována v kódování Native . Nastavení JMS_IBM_ENCODING nemá žádný efekt.

Kód v souboru Obrázek 24 na stránce 172 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. Přijměte JMSBytesMessage pomocí DataInputStream

Kódová stránka se vytiskne pomocí vlastnosti kódové stránky dat vstupní zprávy JMS_IBM_CHARACTER_SET. Na vstupu JMS_IBM_CHARACTER_SET je kódová stránka Java a ne číselný identifikátor kódované znakové sady.

Tabulka typů zpráv a typů převodů

Tabulka 32. Typy zpráv a typy převodů

	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 32. Typy zpráv a typy převodů (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

JMS přístupů k převodu zpráv

Návrhářům aplikací JMS je otevřena řada přístupů pro převod dat. Tyto přístupy nejsou výlučné; některé aplikace pravděpodobně používají kombinaci těchto přístupů. Pokud vaše aplikace pouze mění text nebo si vyměňují zprávy pouze s jinými aplikacemi JMS , obvykle převod dat nepovažujete za možný. Převod dat se provádí automaticky za vás, pomocí IBM MQ.

JMS převod a kódování zpráv klienta

Jsou uvedeny metody, které používáte k provedení převodu a kódování zpráv klienta JMS , s příklady kódu pro každý typ převodu.

Převod dat správce front

Převod dat správce front byl vždy k dispozici pro jiné aplikace nežJMS , které přijímají zprávy od klientů systému JMS . Klienti systému JMS , kteří přijímají zprávy, také používají převod dat správce front, který je volitelný.

Související úlohy

Výměna formátovaného záznamu s jinou aplikací nežJMS

Postupujte podle kroků navržených v této úloze, chcete-li navrhnout a sestavit uživatelskou proceduru pro převod dat a klientskou aplikaci JMS , která si může vyměňovat zprávy s jinou aplikací nežJMS pomocí produktu JMSBytesMessage. K výměně formátované zprávy s jinou aplikací nežJMS může dojít s voláním uživatelské procedury pro převod dat nebo bez volání.

JMS převod a kódování zpráv klienta

Jsou uvedeny metody, které používáte k provedení převodu a kódování zpráv klienta JMS , s příklady kódu pro každý typ převodu.

K převodu a kódování dochází, když jsou primitiva Java nebo objekty čteny nebo zapisovány do zpráv JMS a z nich. Převod se nazývá převod dat klienta JMS , aby se odlišil od převodu dat správce front a převodu dat aplikace. Převod probíhá striktně, když jsou data čtena nebo zapsána do zprávy JMS . Text je převeden do a z vnitřní 16bitové reprezentace Unicode[3]na znakovou sadu používanou pro text ve zprávách. Číselná

data jsou převedena na primitivní číselné typy Java a na kódování definované pro zprávu. Zda se provádí převod a jaký typ převodu se provádí, závisí na typu zprávy JMS a na operaci čtení nebo zápisu.

Tabulka 33 na stránce 174 kategorizuje metody čtení a zápisu pro různé JMS typy zpráv podle typu provedeného převodu. Typy převodů jsou popsány v textu za tabulkou.

Tabulka 33. Typy zpráv a typy převodů

	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] Některé reprezentace Unicode vyžaduje více než 16 bitů. Viz odkaz Java SE.

Text

Výchozí hodnota CodedCharacterSetId pro místo určení je 1208, UTF-8. Standardně je text převeden z Unicode a odeslán jako UTF-8 textový řetězec. Při příjmu je text převeden z kódované znakové sady ve zprávě přijaté klientem do kódování Unicode.

Metody setText a writeString převádějí text z 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 odesílání zprávy musí být použit číselný identifikátor kódované znakové sady.[4].

Úseky kódu v produktu “Odeslání a přijetí JMSTextmessage” na stránce 177 odesílají dvě zprávy. Jeden je odeslán ve znakové sadě definované pro cíl a druhý ve znakové sadě 37, definované aplikací.

Metody getText a readString převádějí text ve zprávě ze znakové sady definované ve zprávě do kódování Unicode. Metody používají kódovou stránku definovanou ve vlastnosti zprávy JMS_IBM_CHARACTER_SET. Kódová stránka je mapována z adresáře MQRFH2.CodedCharacterSetId , pokud zpráva není typu MQa nemá hodnotu MQRFH2. Pokud se jedná o zprávu typu MQbez hodnoty MQRFH2, bude kódová stránka mapována z adresáře MQMD.CodedCharacterSetId.

Úsek kódu v produktu Obrázek 29 na stránce 177 přijme zprávu, která byla odeslána do místa určení. 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 . Projděte frontu a zobrazte vlastnosti zprávy před jejím načtením.

Porovnejte úsek kódu v souboru Obrázek 28 na stránce 177 s nesprávným úsekem kódu v souboru Obrázek 25 na stránce 175. V nesprávném úseku kódu je textový řetězec převeden dvakrát, jednou pomocí aplikace a znovu pomocí 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 má před sebou délku 2 bajty. Maximální délka textového řetězce je 65534 bajtů. Metoda readUTF čte položku ve zprávě napsané metodou writeUTF . Čte přesně počet bajtů zapsaných metodou writeUTF .

Číselné

Výchozí číselné kódování pro cíl je Native. Konstanta kódování Native pro Java má hodnotu 273, x'00000111', která je stejná pro všechny platformy. Při příjmu jsou čísla ve zprávě správně transformována na číselná primitiva Java . Transformace používá kódování definované ve zprávě a typ vrácený metodou čtení.

Metoda send převede čísla přidaná do zprávy pomocí set a write do číselného kódování definovaného pro cíl. Cílové kódování může být pro zprávu přepsáno aplikací nastavující vlastnost zprávy JMS_IBM_ENCODING ; například:

```
message.setIntProperty(WMQConstants.JMS_IBM_ENCODING,
WMQConstants.WMQ_ENCODING_INTEGER_REVERSED);
```

Číselné metody get a read převádí čísla ve zprávě z číselného kódování definovaného ve zprávě. Převádí čísla na typ určený metodou read nebo get ; viz Vlastnost ENCODING. Metody

[4] Při příjmu zprávy JMS_IBM_CHARACTER_SET je uveden název kódové stránky Java Charset .

používají kódování definované v souboru JMS_IBM_ENCODING. Kódování je mapováno z produktu MQRFH2.Encoding , pokud zpráva není typu MQa nemá hodnotu MQRFH2. Pokud se jedná o zprávu typu MQbez hodnoty MQRFH2, budou metody používat kódování definované v souboru MQMD.Encoding.

Příklad v souboru Obrázek 30 na stránce 178 zobrazuje aplikaci kódující číslo v cílovém formátu a odesílající jej v souboru JMSStreamMessage. Porovnejte příklad v souboru Obrázek 30 na stránce 178 s příkladem v souboru Obrázek 31 na stránce 178. Rozdíl je v tom, že JMS_IBM_ENCODING musí být nastaveno v JMSBytesMessage.

Poznámka: Jednoduchým způsobem, jak zkontrolovat, zda je číslo správně zakódováno, je použít Průzkumníka IBM MQ . Procházejte frontu a zobrazte vlastnosti zprávy před jejím spotřebováním.

Jiný

Metody boolean kódují true a false jako x'01' a x'00' v JMSByteMessage, JMSStreamMessagea JMSMapMessage.

Metody UTF kódují a dekódují Unicode do textových řetězců UTF-8 . Řetězce jsou omezeny na méně než 65536 znaků a předchází jim dvoubajtové pole délky.

Metody objektu zalamují primitivní typy jako objekty. Číselné a textové typy jsou kódovány nebo převáděny, jako by primitivní typy byly čteny nebo zapisovány pomocí numerických a textových metod.

Není

Metody readByte, readBytes, readUnsignedByte, writeBytea writeBytes získají nebo vloží jednotlivé bajty nebo pole bajtů mezi aplikaci a zprávu bez převodu. Metody readChar a writeChar získají a vloží 2 bajtové znaky Unicode mezi aplikaci a zprávu bez převodu.

Pomocí metod readBytes a writeBytes může aplikace provádět vlastní převod kódových bodů, jako například “Odesílání a příjem textu v souboru JMSBytesMessage” na stránce 178.

Produkt IBM MQ neprovádí v klientu žádný převod kódové stránky, protože zpráva je JMSBytesMessage, a protože se používají metody readBytes a writeBytes . Nicméně, pokud bajty představují text, ujistěte se, že kódová stránka použitá aplikací odpovídá kódované znakové sadě místa určení. Zpráva může být znovu převedena uživatelskou procedurou pro převod správce front. Další možností je, že přijímající klientský program JMS může postupovat podle konvence převodu všech bajtových polí představujících text ve zprávě na řetězce nebo znaky pomocí vlastnosti JMS_IBM_CHARACTER_SET ve zprávě.

V tomto příkladu klient používá pro převod cílovou kódovanou znakovou sadu:

```
bytes.writeBytes("In the destination code page".getBytes(
CCSID.getCodepage(((MQDestination) destination)
.getIntProperty(WMQConstants.WMQ_CCSID))));
```

Alternativně mohl klient zvolit kódovou stránku a poté nastavit odpovídající kódovanou znakovou sadu ve vlastnosti JMS_IBM_CHARACTER_SET zprávy. Parametr IBM MQ classes for Java použijte JMS_IBM_CHARACTER_SET k nastavení pole CodedCharacterSetId ve vlastnostech JMS v souboru MQRFH2nebo v deskriptoru zprávy MQMD:

```
String codePage = CCSID.getCodepage(37);
message.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET, codePage);
5
```

Pokud je bajtové pole zapsáno do JMSStringMessage nebo JMSMapMessage, IBM MQ classes for JMS neprovede převod dat, protože bajty jsou zapsány jako hexadecimální data, nikoli jako text v JMSStringMessage a JMSMapMessage.

---

[5] SetStringProperty(WMQConstants.JMS_IBM_CHARACTER_SET, codePage) currently accepts only numeric character set identifiers.

Pokud bajty představují znaky ve vaší aplikaci, musíte vzít v úvahu, jaké kódové body se mají číst a zapisovat do zprávy. Kód v souboru Obrázek 26 na stránce 177 se řídí konvencí použití cílové kódované znakové sady. Pokud vytvoříte řetězec pomocí výchozí znakové sady pro prostředí JVM, bude bajtový obsah záviset na platformě. Prostředí JVM v systému Windows má obvykle výchozí hodnotu `Charset` `windows-1252`a AIX and Linux má hodnotu `UTF-8`. Pro výměnu mezi Windowsa AIX and Linuxmusíte vybrat explicitní kódovou stránku pro výměnu textu v bajtech.

```
StreamMessage smo = producer.session.createStreamMessage();
smo.writeBytes("123".getBytes(CCSID.getCodepage(((MQDestination) destination)
.getIntProperty(WMQConstants.WMQ_CCSID))));
```

Obrázek 26. Zápis bajtů reprezentujících řetězec v souboru JMS`StreamMessage` pomocí cílové znakové sady

Příklady

Odeslání a přijetí `JMSTextmessage`

Textová zpráva nemůže obsahovat text v různých znakových sadách. Příklad zobrazuje text v různých znakových sadách, odesílaný 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 cílem

```
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 `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 zobrazují číslo odesílané v kódování definovaném pro cíl. Všimněte si, že musíte nastavit vlastnost `JMS_IBM_ENCODING` položky `JMSBytesMessage` na hodnotu zadanou pro cíl.

```
StreamMessage smo = session.createStreamMessage();
smo.writeInt(256);
producer.send(smo);
...
StreamMessage smi = (StreamMessage)consumer.receive();
System.out.println(smi.readInt());
...
256
```

Obrázek 30. Odeslání čísla pomocí kódování místa určení v souboru 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. Odeslání čísla pomocí kódování místa určení v souboru JMS`BytesMessage`

Odesílání a příjem textu v souboru JMSBytesMessage

Kód v poli Obrázek 32 na stránce 178 odešle řetězec v poli BytesMessage. Pro jednoduchost příklad odešle jeden řetězec, pro který je vhodnější JMS`TextMessage` . Chcete-li přijmout textový řetězec v bajtech obsahující směs typů, musíte znát délku řetězce v bajtech s názvem *TEXT_LENGTH* in Obrázek 33 na stránce 178. I pro řetězec s pevným počtem znaků může být délka bajtové reprezentace 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 souboru `String` v adresáři JMS`BytesMessage`

```
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 JMS`BytesMessage`

Související pojmy

JMS přístupů k převodu zpráv

Návrhářům aplikací JMS je otevřena řada přístupů pro převod dat. Tyto přístupy nejsou výlučné; některé aplikace pravděpodobně používají kombinaci těchto přístupů. Pokud vaše aplikace pouze mění text nebo si vyměňují zprávy pouze s jinými aplikacemi JMS , obvykle převod dat nepovažujete za možný. Převod dat se provádí automaticky za vás, pomocí IBM MQ.

Převod dat správce front

Převod dat správce front byl vždy k dispozici pro jiné aplikace nežJMS , které přijímají zprávy od klientů systému JMS . Klienti systému JMS , kteří přijímají zprávy, také používají převod dat správce front, který je volitelný.

Související úlohy

Výměna formátovaného záznamu s jinou aplikací nežJMS

Postupujte podle kroků navržených v této úloze, chcete-li navrhnout a sestavit uživatelskou proceduru pro převod dat a klientskou aplikaci JMS , která si může vyměňovat zprávy s jinou aplikací nežJMS pomocí produktu `JMSBytesMessage`. K výměně formátované zprávy s jinou aplikací nežJMS může dojít s voláním uživatelské procedury pro převod dat nebo bez volání.

Související odkazy

JMS typy zpráv a převod

Volba typu zprávy ovlivňuje váš přístup k převodu zpráv. Interakce převodu zpráv a typu zpráv je popsána pro JMS typy zpráv `JMSObjectMessage`, `JMSTextMessage`, `JMSMapMessage`, `JMSStreamMessage`a `JMSBytesMessage`.

Převod dat správce front

Převod dat správce front byl vždy k dispozici pro jiné aplikace nežJMS , které přijímají zprávy od klientů systému JMS . Klienti systému JMS , kteří přijímají zprávy, také používají převod dat správce front, který je volitelný.

Správce front může převádět znaková a číselná data v datech zprávy pomocí hodnot `CodedCharacterSetId`, `Encoding`a `Format` nastavených pro data zprávy. Pro jiné aplikace nežJMS byla možnost převodu vždy k dispozici nastavením volby `GetMessage`, `GMO_CONVERT`.

Správce front může převádět zprávy odeslané klientům JMS . Převod správce front je řízen nastavením vlastnosti místa určení `WMQ_RECEIVE_CONVERSION`na hodnotu `WMQ_RECEIVE_CONVERSION_QMGR`nebo `WMQ_RECEIVE_CONVERSION_CLIENT_MSG`. Aplikace může změnit nastavení cíle:

```
((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 standardně transformována do UTF-8 (1208). Následné metody read a get dekódují text v přijatých datech z UTF-8, čímž vytvoří textová primitiva Java v jejich interním kódování Unicode. UTF-8 není jedinou cílovou znakovou sadou z převodu dat správce front. Můžete zvolit jiný CCSID nastavením vlastnosti cíle `WMQ_RECEIVE_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 kódovanou znakovou sadu pro převod správce front

Příčina změny `WMQ_RECEIVE_CCSID` je specializovaná. Zvolený CCSID se neliší od textových objektů vytvořených v prostředí JVM. Avšak některá prostředí JVM na některých platformách 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 platformy klienta JMS měly problémy s doručováním textu zprávy v kódování UTF-8.

Kód JMS je ekvivalentní tučnému textu v kódu C v souboru Obrázek 36 na stránce 180,

```
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 formát definovaný uživatelem, 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

JMS přístupů k převodu zpráv
Návrhářům aplikací JMS je otevřena řada přístupů pro převod dat. Tyto přístupy nejsou výlučné; některé aplikace pravděpodobně používají kombinaci těchto přístupů. Pokud vaše aplikace pouze mění text nebo si vyměňují zprávy pouze s jinými aplikacemi JMS , obvykle převod dat nepovažujete za možný. Převod dat se provádí automaticky za vás, pomocí IBM MQ.

JMS převod a kódování zpráv klienta
Jsou uvedeny metody, které používáte k provedení převodu a kódování zpráv klienta JMS , s příklady kódu pro každý typ převodu.

“Vyvolání uživatelské procedury pro převod dat” na stránce 951

Uživatelská procedura pro převod dat je uživatelská procedura, která přijímá řízení během zpracování volání MQGET.

Související úlohy

Výměna formátovaného záznamu s jinou aplikací nežJMS
Postupujte podle kroků navržených v této úloze, chcete-li navrhnout a sestavit uživatelskou proceduru pro převod dat a klientskou aplikaci JMS , která si může vyměňovat zprávy s jinou aplikací nežJMS pomocí produktu `JMSBytesMessage`. K výměně formátované zprávy s jinou aplikací nežJMS může dojít s voláním uživatelské procedury pro převod dat nebo bez volání.

Související odkazy

JMS typy zpráv a převod
Volba typu zprávy ovlivňuje váš přístup k převodu zpráv. Interakce převodu zpráv a typu zpráv je popsána pro JMS typy zpráv `JMSObjectMessage`, `JMSTextMessage`, `JMSMapMessage`, `JMSStreamMessage`a `JMSBytesMessage`.

Výměna formátovaného záznamu s jinou aplikací nežJMS
Postupujte podle kroků navržených v této úloze, chcete-li navrhnout a sestavit uživatelskou proceduru pro převod dat a klientskou aplikaci JMS , která si může vyměňovat zprávy s jinou aplikací nežJMS pomocí produktu `JMSBytesMessage`. K výměně formátované zprávy s jinou aplikací nežJMS může dojít s voláním uživatelské procedury pro převod dat nebo bez volání.

Než začnete

Můžete být schopni navrhnout jednodušší řešení pro výměnu zpráv s jinou aplikací nežJMS pomocí produktu `JMSTextMessage`. Před provedením kroků v této úloze tuto možnost eliminujte.

Informace o této úloze

Klient JMS se snadněji zapisuje, pokud se nepodílí na podrobnostech formátování JMS zpráv vyměňovaných s ostatními klienty JMS . Dokud je typ zprávy `JMSTextMessage`, `JMSMapMessage`, `JMSStreamMessage`nebo `JMSObjectMessage`, IBM MQ se stará o podrobnosti formátování zprávy. Produkt IBM MQ se zabývá rozdíly v kódových stránkách a číselném kódování na různých platformách.

Tyto typy zpráv můžete použít k výměně zpráv s jinými aplikacemi nežJMS . Chcete-li tak učinit, musíte pochopit, jak jsou tyto zprávy vytvářeny produktem IBM MQ classes for JMS. Můžete být schopni upravit aplikaci, která neníJMS , tak, aby interpretovala zprávy; viz “Mapování zpráv JMS na zprávy IBM MQ” na stránce 144.

Výhodou použití jednoho z těchto typů zpráv je, že programování klienta JMS nezávisí na typu aplikace, se kterou si vyměňuje zprávy. Nevýhodou je, že může vyžadovat úpravu jiného programu a vy nemusíte být schopni tento jiný program změnit.

Alternativním přístupem je napsat aplikaci klienta JMS , která může pracovat s existujícími formáty zpráv. Často existující zprávy mají pevný formát a obsahují směs neformátovaných dat, textu a čísel. Jako výchozí bod pro sestavení klienta JMS , který si může vyměňovat formátované záznamy s jinými aplikacemi nežJMS , použijte kroky v této úloze a příklad JMS klienta v produktu “Zápis tříd pro zapouzdření rozvržení záznamu v JMSBytesMessage” na stránce 184.

Postup

1. Definujte rozvržení záznamu nebo použijte jednu z předdefinovaných tříd záhlaví IBM MQ .

   Chcete-li zacházet s předdefinovanými záhlavími IBM MQ , prohlédněte si téma Obsluha IBM MQ záhlaví zpráv.

   Obrázek 37 na stránce 182 je příkladem uživatelem definovaného rozvržení záznamu s pevnou délkou, který může být zpracován obslužným programem pro převod dat.
2. Vytvořte uživatelskou proceduru pro převod dat.

   Chcete-li napsat uživatelskou proceduru pro převod dat, postupujte podle pokynů v části Psaní uživatelského programu pro převod dat .

Chcete-li vyzkoušet příklad v části “Zápis tříd pro zapouzdření rozvržení záznamu v JMSBytesMessage” na stránce 184, pojmenujte uživatelskou proceduru pro převod dat MYRECORD.

3. Zapište třídy Java , abyste zapouzdřili rozvržení záznamu a odeslali a přijali záznam. Dva přístupy, které můžete mít, jsou:
   - Zapište třídu, která čte a zapíše JMSBytesMessage , který obsahuje záznam; viz “Zápis tříd pro zapouzdření rozvržení záznamu v JMSBytesMessage” na stránce 184.
   - Zapište rozšíření třídy com.ibm.mq.header.Header , abyste definovali datovou strukturu záznamu; viz Vytvoření tříd pro nové typy záhlaví.
4. Rozhodněte, v jaké kódované znakové sadě se mají zprávy vyměňovat.

   Viz Výběr přístupu k převodu zpráv: přijímač je dobrý.
5. Konfigurujte místo určení pro výměnu zpráv typu MQbez záhlaví JMS MQRFH2 .

   Cíl odesílání i cíl příjmu musí být nakonfigurovány tak, aby si vyměňovaly zprávy typu MQ. Pro odesílání i příjem můžete použít stejný cíl.

   Aplikace může přepsat vlastnost těla cílové zprávy:

   ```
   ((MQDestination)destination).setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_MQ);
   ```

   Příklad v souboru “Zápis tříd pro zapouzdření rozvržení záznamu v JMSBytesMessage” na stránce 184 potlačí vlastnost těla cílové zprávy a zajistí odeslání zprávy ve stylu MQ.
6. Otestujte řešení pomocí aplikací JMS a jiných aplikací nežJMS .

   Užitečné nástroje pro testování uživatelské procedury převodu dat jsou:
   - Ukázkový program amqsgetc0.c je užitečný pro testování přijetí zprávy odeslané klientem JMS . Viz navrhované úpravy pro použití ukázkového záhlaví RECORD.h v souboru Obrázek 38 na stránce 183. S úpravami produkt amqsgetc0.c obdrží zprávu odeslanou ukázkovým klientem JMS , TryMyRecord.java ; viz “Zápis tříd pro zapouzdření rozvržení záznamu v JMSBytesMessage” na stránce 184.
   - Ukázkový IBM MQ program pro procházení amqsbcg0.cje užitečný pro kontrolu obsahu záhlaví zprávy, záhlaví JMS , MQRFH2a obsahu zprávy.
   - Program **rfhutil** , dříve dostupný v SupportPac IH03, umožňuje zachycení a uložení testovacích zpráv v souborech a jejich následné použití k řízení toků zpráv. Výstupní zprávy lze také číst a zobrazovat v různých formátech. Tyto formáty zahrnují dva typy XML, stejně jako porovnání s zakladači v jazyce COBOL. Data mohou být v EBCDIC nebo ASCII. Před odesláním zprávy lze do zprávy přidat záhlaví RFH2 .

   Pokud se pokusíte přijmout zprávy pomocí upraveného ukázkového programu amqsgetc0.c a obdržíte chybu s kódem příčiny 2080, zkontrolujte, zda zpráva obsahuje MQRFH2. Úpravy předpokládají, že zpráva byla odeslána do místa určení, které neuvádí hodnotu MQRFH2.

Příklady

```
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

- Deklarujte datovou strukturu 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);
```

- Upravit volání MQGET pro použití 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           */
```

- Změňte příkaz print,

  1. Od:

```
buffer[messlen] = '\0';            /* add terminator          */
printf("message <%s>\n", buffer);
```

  2. Komu:

```
/* 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

JMS přístupů k převodu zpráv

Návrhářům aplikací JMS je otevřena řada přístupů pro převod dat. Tyto přístupy nejsou výlučné; některé aplikace pravděpodobně používají kombinaci těchto přístupů. Pokud vaše aplikace pouze mění text nebo si vyměňují zprávy pouze s jinými aplikacemi JMS , obvykle převod dat nepovažujete za možný. Převod dat se provádí automaticky za vás, pomocí IBM MQ.

JMS převod a kódování zpráv klienta

Jsou uvedeny metody, které používáte k provedení převodu a kódování zpráv klienta JMS , s příklady kódu pro každý typ převodu.

Převod dat správce front
Převod dat správce front byl vždy k dispozici pro jiné aplikace nežJMS , které přijímají zprávy od klientů systému JMS . Klienti systému JMS , kteří přijímají zprávy, také používají převod dat správce front, který je volitelný.

Obslužný program pro vytvoření kódu ukončení převodu

Související odkazy

JMS typy zpráv a převod
Volba typu zprávy ovlivňuje váš přístup k převodu zpráv. Interakce převodu zpráv a typu zpráv je popsána pro JMS typy zpráv `JMSObjectMessage`, `JMSTextMessage`, `JMSMapMessage`, `JMSStreamMessage`a `JMSBytesMessage`.

Zápis tříd pro zapouzdření rozvržení záznamu v `JMSBytesMessage`
Účelem této úlohy je například prozkoumat, jak kombinovat převod dat a rozvržení pevných záznamů v produktu `JMSBytesMessage`. V úloze vytvoříte některé třídy Java pro výměnu vzorové struktury záznamu v `JMSBytesMessage`. Můžete upravit příklad, chcete-li zapsat třídy pro výměnu jiných struktur záznamů.

`JMSBytesMessage` je nejlepší volbou typu zprávy JMS pro výměnu záznamů smíšených datových typů s programy, které nejsou typuJMS . Nemá žádná další data vložená poskytovatelem JMS do těla zprávy. Jedná se proto o nejlepší volbu typu zprávy, kterou lze použít v případě, že klientský program JMS spolupracuje s existujícím programem IBM MQ . Hlavní výzva při použití `JMSBytesMessage` přichází s odpovídající kódování a znakovou sadu očekávanou jiným programem. Řešením je vytvořit třídu, která zapouzdří záznam. Třída, která zapouzdřuje čtení a zápis `JMSBytesMessage`pro specifický typ záznamu, usnadňuje odesílání a přijímání záznamů v pevném formátu v programu JMS . Zachycením generických aspektů rozhraní v abstraktní třídě lze většinu řešení znovu použít pro různé formáty záznamů. 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`, pro sestavení formátu záznamu deklarativnějším způsobem. Nevýhodou použití třídy `Header` je získání a nastavení atributů používá složitější interpretační rozhraní. Oba přístupy vedou ke stejnému množství kódu aplikace.

`JMSBytesMessage` může kromě formátu `MQRFH2`zapouzdřit pouze jeden formát 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 `MQRFH2`. Příklad je napsán s předpokladem, že `JMSBytesMessage` obsahuje pouze jeden záznam uživatele.

Než začnete

1. Vaše úroveň dovedností: musíte být obeznámeni s Java programováním a JMS. Nejsou poskytnuty žádné pokyny pro nastavení vývojového prostředí Java . Je výhodné mít napsaný program pro výměnu `JMSTextMessage`, `JMSStreamMessage`nebo `JMSMapMessage`. Pak můžete vidět rozdíly ve výměně zprávy pomocí `JMSBytesMessage`.
2. Příklad vyžaduje IBM WebSphere MQ 7.0.
3. Příklad byl vytvořen pomocí perspektivy Java pracovní plochy Eclipse . Vyžaduje prostředí JRE 6.0 nebo vyšší. Pomocí perspektivy Java v Průzkumníku IBM MQ můžete vyvíjet a spouštět třídy Java . Případně použijte vlastní vývojové prostředí Java .
4. Použití produktu IBM MQ Explorer usnadňuje nastavení testovacího prostředí a ladění, než použití obslužných programů příkazového řádku.

Informace o této úloze

Budete provedeni vytvořením dvou tříd: `RECORD` a `MyRecord`. Tyto dvě třídy společně zapouzdřují záznam v pevném formátu. Mají metody pro získání a nastavení atributů. Metoda get přečte záznam z `JMSBytesMessage` a metoda put zapíše záznam do `JMSBytesMessage`.

Účelem úlohy není vytvořit produkční třídu kvality, kterou můžete znovu použít. Můžete se rozhodnout použít příklady v úloze, abyste mohli začít používat své vlastní třídy. Účelem úlohy je poskytnout vám vodicí poznámky, především o použití znakových sad, formátů a kódování, při použití JMSBytesMessage. Každý krok při vytváření tříd je vysvětlen a jsou popsány aspekty použití produktu JMSBytesMessage, které jsou někdy přehlíženy.

Třída RECORD je abstraktní a definuje některá společná pole pro záznam uživatele. Obecná pole jsou modelována podle standardního rozvržení záhlaví IBM MQ , které obsahuje poutač, verzi a délku pole. Pole kódování, znaková sada a formát, která se nacházejí v mnoha záhlavích IBM MQ , jsou vynechána. Jiné záhlaví nemůže sledovat formát definovaný uživatelem. Třída MyRecord , která rozšiřuje třídu RECORD , tak činí doslovným rozšířením záznamu o další pole uživatele. Položku JMSBytesMessagevytvořenou třídami lze zpracovat uživatelskou procedurou pro převod dat správce front.

"Třídy použité ke spuštění příkladu" na stránce 191 obsahuje úplný seznam položek RECORD a MyRecord. Zahrnuje také výpisy dalších tříd "lešení" pro testování RECORD a MyRecord. Další třídy jsou:

TryMyRecord
Hlavní program pro testování RECORD a MyRecord.

EndPoint
Abstraktní třída, která zapouzdřuje připojení JMS , cíl a relaci do jediné třídy. Jeho rozhraní pouze splňuje potřeby testování tříd RECORD a MyRecord . Nejedná se o zavedený vzor návrhu pro psaní aplikací JMS .

Poznámka: Třída Endpoint zahrnuje tento řádek kódu po vytvoření cíle:

```
((MQDestination)destination).setReceiveConversion
                            (WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
```

V produktu V7.0je v produktu V7.0.1.5nutné zapnout převod správce front. Podle výchozího nastavení je tato volba zakázána. V produktu V7.0je standardně povolen převod správce front V7.0.1.4 a tento řádek kódu způsobuje chybu.

MyProducer a MyConsumer
Třídy, které rozšiřují EndPointa vytvářejí MessageConsumer a MessageProducer, připojené a připravené přijímat požadavky.

Všechny třídy dohromady tvoří úplnou aplikaci, se kterou můžete sestavit a experimentovat, abyste pochopili, jak používat převod dat v systému JMSBytesMessage.

Postup

1. Vytvořte abstraktní třídu pro zapouzdření standardních polí v záhlaví IBM MQ s výchozím konstruktorem. Později rozšíříte třídu tak, aby přizpůsobila 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` to `nextFormat`jsou uvedeny v pořadí, v jakém jsou uspořádány 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 kódovaných znakových sad. Vzorový kód používá znakové sady.

d. `int` je serializován na `MQLONG` pomocí IBM MQ. `MQLONG` je 4 bajty.

2. Vytvořte metody 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. Položky `messageCharset` a `messageEncoding`jsou zachyceny z vlastností zprávy, protože potlačují hodnoty nastavené pro místo určení. Soubor `format` není aktualizován. Příklad neprovádí kontrolu chyb. Je-li volán konstruktor `Record(BytesMessage)` , předpokládá se, že `JMSBytesMessage` je zpráva typu `RECORD` . Řádek "`setStructID(new String(structID, getMessageCharset()))`" nastaví zvýrazňovač.

b. Řádky kódu, které vyplňují deserializovaná pole metody ve zprávě, v pořadí aktualizace výchozích hodnot nastavených v instanci RECORD.

4. Vytvořte metodu vložení 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 do jedné třídy. MyConsumer, použitý později, zapouzdřuje JMS Connection, Destination, Sessiona MessageConsumer do jedné třídy.

b. V případě JMSBytesMessage, je-li kódování jiné než Native, musí být kódování nastaveno ve zprávě. Cílové kódování se zkopíruje do atributu kódování zpráv JMS_IBM_CHARACTER_SETa uloží se jako atribut třídy RECORD .

   i) "setMessageEncoding(myProducer.getEncoding());" volá "(((MQDestination) destination).getIntProperty(WMQConstants.WMQ_ENCODING));" k získání cílového kódování.

   ii) "Bytes.setIntProperty(WMQConstants.JMS_IBM_ENCODING, getMessageEncoding());" nastavuje kódování zpráv.

c. Znaková sada použitá k transformaci textu na bajty je získána z místa určení a uložena jako atribut třídy RECORD . Není nastaven ve zprávě, protože není používán IBM MQ classes for JMS při zápisu JMSBytesMessage.

   "messageCharset = myProducer.getCharset();" volání

```
    public String getCharset() throws UnsupportedEncodingException,
            JMSException {
        return CCSID.getCodepage(getCCSID());
}
```

   Získá znakovou sadu Java z identifikátoru kódované znakové sady.

   " CCSID.getCodepage(ccsid) " je v balíku com.ibm.mq.headers. ccsid je získán z jiné metody v souboru MyProducer, která se dotazuje na místo určení:

```
    public int getCCSID() throws JMSException {
        return (((MQDestination) destination)
                .getIntProperty(WMQConstants.WMQ_CCSID));
    }
```

d. "myProducer.setMQClient(true);" přepíše nastavení místa určení pro typ klienta a vynutí jej na IBM MQ MQI client. Můžete raději vynechat tento řádek kódu, protože zakrývá administrativní chybu konfigurace.

   "myProducer.setMQClient(true);" volání:

```
((MQDestination) destination).setTargetClient(WMQConstants.WMQ_TARGET_DEST_MQ); }
if (!getMQDest()) setMQBody();
```

Kód má vedlejší účinek nastavení stylu těla IBM MQ na nespecifikovaný, pokud musí přepsat nastavení JMS.

Poznámka:

IBM MQ classes for JMS zapíše formát, kódování a identifikátor znakové sady zprávy do deskriptoru zprávy 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 JMS_IBM_MQMD_* . Chcete-li nastavit vlastnosti JMS_IBM_MQMD_* , musíte nastavit vlastnost cíle WMQ_MQMD_WRITE_ENABLED :

```
((MQDestination)destination).setMQMDWriteEnabled(true);
```

Chcete-li číst vlastnosti, musíte nastavit vlastnost cíle WMQ_MQMD_READ_ENABLED.

Volbu JMS_IBM_MQMD_* použijte pouze v případě, že máte úplnou kontrolu nad celým informačním obsahem zprávy. Na rozdíl od vlastností JMS_IBM_* vlastnosti JMS_IBM_MQMD_* neřídí, jak IBM MQ classes for JMS vytváří zprávu JMS . Je možné vytvořit vlastnosti deskriptoru zpráv, které jsou v konfliktu s vlastnostmi zprávy JMS .

e. Řádky kódu, které dokončí metodu, serializují atributy ve třídě jako pole ve zprávě.

Řetězcové atributy jsou vyplněny mezerami. Řetězce jsou převedeny na bajty pomocí znakové sady definované pro záznam a zkráceny 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 jakarta.jms.BytesMessage;
import jakarta.jms.JMSException;
import com.ibm.mq.constants.MQConstants;
import com.ibm.mq.headers.MQDataException;
import com.ibm.msg.client.wmq.WMQConstants;
```

6. Vytvořte třídu pro rozšíření třídy RECORD o 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 zachytovač očí, 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 nejprve přečtena třídou RECORD .

b. Text recordData je převeden na String pomocí vlastnosti znakové sady zprávy.

9. Vytvořte statickou metodu pro získání zprávy od spotřebitele a vytvořte novou instanci MyRecord .

```
public static MyRecord get(MyConsumer myConsumer) throws JMSException,
       MQDataException, IOException {
   BytesMessage message = (BytesMessage) myConsumer.receive();
   return new MyRecord(message);
}
```

Poznámka:

a. V příkladu je pro stručnost volán konstruktor MyRecord(BytesMessage) ze statické metody get. Obvykle můžete oddělit příjem zprávy od vytvoření nové instance MyRecord .

10. Vytvořte metodu vložení pro připojení polí zákazníka k souboru 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. Volání metody v kódu serializují atributy ve třídě MyRecord jako pole ve zprávě.

- Atribut recordData String je doplněn mezerami, převeden na bajty pomocí znakové sady definované pro záznam a zkrácen na délku polí RecordData .

11. Dokončete třídu přidáním příkazů include.

```
 package com.ibm.mq.id;
import java.io.IOException;
import jakarta.jms.BytesMessage;
import jakarta.jms.JMSException;
```

```
import com.ibm.mq.headers.MQDataException;
```

Výsledky

- Výsledky spuštění třídy `TryMyRecord` :
  - Odeslání zprávy v kódované znakové sadě 37 a 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
```

  - Odeslání zprávy v kódované znakové sadě 37 a nepoužití 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 546 CCSID IBM037
```

- Výsledkem úpravy třídy `TryMyRecord` není přijetí zprávy a její přijetí pomocí upravené ukázky `amqsget0.c` . Upravená ukázka přijímá formátovaný záznam; viz Obrázek 38 na stránce 183 v “Výměna formátovaného záznamu s jinou aplikací nežJMS” na stránce 181.
  - Odeslání zprávy v kódované znakové sadě 37 a použití uživatelské procedury pro převod správce front:

```
Sample AMQSGET0 start
ccsid <850>, flags <1>, message <ABCDEFGHIJKLMNOPQRSTUVWXYZ012345>
no more messages
Sample AMQSGET0 end
```

  - Odeslání zprávy v kódované znakové sadě 37 a nepoužití uživatelské procedury 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 uživatelskou procedurou pro převod dat. Vytvořte třídy Java , nakonfigurujte IBM MQa spusťte hlavní program `TryMyRecord` ; viz “#unique_196/unique_196_Connect_42_Try” na stránce 191.

1. Nakonfigurujte IBM MQ a JMS pro spuštění příkladu. Pokyny jsou pro spuštění příkladu na systému 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 rozhraní JNDI

```
cd c:\
md JNDI-Directory
```

d. Přejděte do adresáře JMS bin.

   Odtud musí být spuštěn administrační program 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 JMS .

```
JMSAdmin < JMSQM1Q1.txt
```

2. Můžete vytvářet, měnit a procházet definice, které jste vytvořili pomocí Průzkumníku IBM MQ .
3. Spusťte příkaz TryMyRecord.

Třídy použité ke spuštění příkladu

Třídy uvedené v následujících blocích kódu jsou také k dispozici v komprimovaném souboru. Stáhněte si soubor jm25529_.zip nebo jm25529_.tar.gz.

TryMyRecord

```
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());
    }
}
```

RECORD

V 9.3.0 JM 3.0 V 9.3.0

```
package com.ibm.mq.id;
import java.io.IOException;
import java.io.Serializable;
import java.io.UnsupportedEncodingException;
import jakarta.jms.BytesMessage;
import jakarta.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; }
}
```

JMS 2.0

```
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; }
}
```

MyRecord

V 9.3.0 JM 3.0 V 9.3.0

```
package com.ibm.mq.id;
import java.io.IOException;
import jakarta.jms.BytesMessage;
import jakarta.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; }
}
```

JMS 2.0

```
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; }
}
```

EndPoint

V 9.3.0 JM 3.0 V 9.3.0

```
package com.ibm.mq.id;
import java.io.UnsupportedEncodingException;
import jakarta.jms.Connection;
import jakarta.jms.ConnectionFactory;
import jakarta.jms.Destination;
import jakarta.jms.JMSException;
import jakarta.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); }
    }
```

JMS 2.0

```
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); }
    }
```

MyProducer

V 9.3.0 JM 3.0 V 9.3.0

```
package com.ibm.mq.id;
import jakarta.jms.JMSException;
import jakarta.jms.Message;
import jakarta.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); }
}
```

JMS 2.0

```
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); }
}
```

MyConsumer

V 9.3.0 JM 3.0 V 9.3.0

```
package com.ibm.mq.id;
import jakarta.jms.JMSException;
import jakarta.jms.Message;
```

```
import jakarta.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();  }
}
```

JMS 2.0

```
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();  }
}
```

Vytvoření a konfigurace továren připojení a míst určení

Aplikace IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging může vytvářet továrny připojení a místa určení jejich načtením jako spravovaných objektů z oboru názvů rozhraní JNDI (Java Naming and Directory Interface) pomocí rozšíření IBM JMS nebo pomocí rozšíření IBM MQ JMS . Aplikace může také použít rozšíření produktu IBM JMS nebo rozšíření produktu IBM MQ JMS k nastavení vlastností továren připojení a míst určení.

Továrny připojení a cíle jsou počáteční body v toku logiky aplikace JMS nebo Jakarta Messaging . Aplikace používá objekt ConnectionFactory k vytvoření připojení k serveru systému zpráv a používá objekt Queue nebo Topic jako cíl pro odesílání zpráv nebo zdroj, ze kterého mají být zprávy přijímány. Aplikace proto musí vytvořit alespoň jednu továrnu připojení a jeden nebo více cílů. Po vytvoření továrny připojení nebo místa určení může být nutné konfigurovat objekt nastavením jedné nebo více jeho vlastností.

V souhrnu může aplikace vytvářet a konfigurovat továrny připojení a místa určení následujícími způsoby:

Použití rozhraní JNDI k načtení spravovaných objektů

Administrátor může použít nástroj pro administraci produktu IBM MQ JMS , jak je popsáno v tématu Konfigurace objektů systému zpráv JMS a Jakarta pomocí administračních nástrojů, nebo IBM MQ Explorer , jak je popsáno v tématu Konfigurace objektů JMS 2.0 pomocí produktu IBM MQ Explorer, k vytvoření a konfiguraci továren připojení a míst určení jako spravovaných objektů v oboru názvů JNDI. Aplikace pak může načíst spravované objekty z oboru názvů rozhraní JNDI. Po načtení spravovaného objektu může aplikace v případě potřeby nastavit nebo změnit jednu nebo více svých vlastností buď pomocí rozšíření produktu IBM JMS , nebo pomocí rozšíření produktu IBM MQ JMS .

Poznámka: V 9.3.0 JM 3.0 V 9.3.0 Pro Jakarta Messaging 3.0nemůžete spravovat rozhraní JNDI pomocí IBM MQ Explorer. Administrace rozhraní JNDI je podporována Jakarta Messaging 3.0 variantou **JMSAdmin**, což je **JMS30Admin**.

Použití rozšíření IBM JMS

Aplikace může používat rozšíření produktu IBM JMS k dynamickému vytváření továren připojení a míst určení za běhu. Aplikace nejprve vytvoří objekt továrny JmsFactorya poté použije metody tohoto

objektu k vytvoření továren připojení a míst určení. Po vytvoření továrny připojení nebo místa určení může aplikace k nastavení svých vlastností použít metody zděděné z rozhraní kontextu JmsProperty. Alternativně může aplikace při vytváření místa určení použít identifikátor URI (Uniform Resource Identifier) k určení jedné nebo více vlastností místa určení.

Použití rozšíření IBM MQ JMS

Aplikace může také používat rozšíření produktu IBM MQ JMS k dynamickému vytváření továren připojení a míst určení za běhu. Aplikace používá dodané konstruktory k vytvoření továren připojení a míst určení. Po vytvoření továrny připojení nebo místa určení může aplikace použít metody objektu k nastavení vlastností. Alternativně může aplikace při vytváření místa určení použít identifikátor URI k určení jedné nebo více vlastností místa určení.

Související úlohy

Konfigurace prostředků systému zpráv JMS a Jakarta

Použití rozhraní JNDI k načtení spravovaných objektů v aplikaci JMS nebo Jakarta Messaging

Chcete-li načíst spravované objekty z oboru názvů rozhraní JNDI (Java Naming and Directory Interface), musí aplikace JMS nebo Jakarta Messaging vytvořit počáteční kontext a poté pomocí metody lookup () načíst objekty.

Aby mohla aplikace načíst spravované objekty z oboru názvů rozhraní JNDI, musí administrátor nejprve vytvořit spravované objekty.

JMS 2.0 Pro systém JMS 2.0může administrátor použít nástroj pro administraci produktu IBM MQ JMS , **JMSAmin**nebo IBM MQ Explorer k vytvoření a údržbě spravovaných objektů v oboru názvů JNDI. Další informace naleznete v tématu Konfigurace továren připojení a míst určení v oboru názvů rozhraní JNDI.

V 9.3.0 JM 3.0 V 9.3.0 Pro Jakarta Messaging 3.0nemůžete spravovat rozhraní JNDI pomocí IBM MQ Explorer. Administrace rozhraní JNDI je podporována Jakarta Messaging 3.0 variantou **JMSAdmin**, což je **JMS30Admin**.

Aplikační server obvykle poskytuje vlastní úložiště pro spravované objekty a vlastní nástroje pro vytváření a údržbu objektů.

Chcete-li načíst spravované objekty z oboru názvů JNDI, musí aplikace nejprve vytvořit počáteční kontext, jak ukazuje následující příklad:

V 9.3.0 JM 3.0 V 9.3.0

```
import jakarta.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);
```

JMS 2.0

```
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
Lokátor jednotného prostředku (URL) adresářové služby. 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

icf
Název třídy továrny počátečního kontextu, která 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 chybu LDAP 84. Chcete-li tento problém vyřešit, vložte před volání funkce InitialDirContext () následující řádek kódu:

```
environment.put(Context.REFERRAL, "throw");
```

Po získání počátečního kontextu může aplikace načíst spravované objekty z oboru názvů rozhraní JNDI pomocí metody lookup (), jak ukazuje následující příklad:

```
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 založeného na protokolu LDAP:

- Objekt ConnectionFactory svázaný s názvem myCF .
- Objekt fronty svázaný s názvem myQ .
- Objekt tématu svázaný s názvem myT

Další informace o použití rozhraní JNDI naleznete v dokumentaci k rozhraní JNDI poskytované společností Oracle Corporation.

Související úlohy
Konfigurace objektů rozhraní JMS 2.0 pomocí Průzkumníka IBM MQ
Konfigurace objektů systému zpráv JMS a Jakarta pomocí nástrojů pro administraci
Konfigurace prostředků rozhraní JMS 2.0 v produktu WebSphere Application Server

Použití rozšíření IBM JMS
IBM MQ classes for JMS (JMS 2.0) a IBM MQ classes for Jakarta Messaging (Jakarta Messaging 3.0) obsahují funkčně identickou sadu rozšíření rozhraní API JMS nazvanou rozšíření IBM JMS . Aplikace může tato rozšíření použít k dynamickému vytváření továren připojení a míst určení za běhu a k nastavení vlastností objektů IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging . Rozšíření lze použít s libovolným poskytovatelem systému zpráv.

Rozšíření produktu IBM JMS jsou sadou rozhraní a tříd v následujících balících:

- com.ibm.msg.client.jms
- com.ibm.msg.client.services

Pro systém Jakarta Messaging 3.0jsou tyto balíky v adresáři `com.ibm.jakarta.client.jar`.

JMS 2.0 Pro systém JMS 2.0jsou tyto balíky v adresáři `com.ibm.mqjms.jar` nebo `com.ibm.mq.allclient.jar`.

Tato rozšíření poskytují následující funkci:

- Mechanizmus založený na továrně pro dynamické vytváření továren připojení a míst určení za běhu namísto jejich načítání jako spravovaných objektů z oboru názvů rozhraní JNDI (Java Naming and Directory Interface).
- Sada metod pro nastavení vlastností objektů IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging
- Sada tříd výjimek s metodami pro získání podrobných informací o problému
- Sada metod pro řízení trasování
- Sada metod pro získání informací o verzi produktu IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging

Pro dynamické vytváření továren připojení a míst určení za běhu a nastavení a získávání jejich vlastností poskytují rozšíření produktu IBM JMS alternativní sadu rozhraní pro rozšíření produktu IBM MQ JMS . Ačkoli jsou rozšíření produktu IBM MQ JMS specifická pro poskytovatele systému zpráv IBM MQ , rozšíření produktu IBM JMS nejsou specifická pro produkt IBM MQ a lze je použít s jakýmkoli poskytovatelem systému zpráv v rámci vrstvené architektury popsané v tématu IBM MQ pro architekturu JMS.

Rozhraní com.ibm.msg.client.wmq.WMQConstants (JMS 2.0) nebo com.ibm.msg.jakarta.client.wmq.WMQConstants (Jakarta Messaging 3.0) obsahuje definice konstant, které může aplikace použít při nastavení vlastností objektů IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging pomocí rozšíření IBM JMS . Rozhraní obsahuje konstanty pro poskytovatele systému zpráv IBM MQ a konstanty JMS , které jsou nezávislé na jakémkoli poskytovateli systému zpráv.

Následující příklady kódu předpokládají, že následující příkazy importu jsou zahrnuty do třídy Java :

V 9.3.0 JM 3.0 V 9.3.0

```
import com.ibm.msg.jakarta.client.jms.*;
import com.ibm.msg.jakarta.client.services.*;
import com.ibm.msg.jakarta.client.wmq.WMQConstants;
```

JMS 2.0

```
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

Aby mohla aplikace vytvářet továrny připojení a místa určení 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 ukazuje následující příklad:

V 9.3.0 JM 3.0 V 9.3.0

```
JmsFactoryFactory ff = JmsFactoryFactory.getInstance(JmsConstants.JAKARTA_WMQ_PROVIDER);
```

JMS 2.0

```
JmsFactoryFactory ff = JmsFactoryFactory.getInstance(JmsConstants.WMQ_PROVIDER);
```

Parametr ve volání getInstance() je konstanta, která identifikuje poskytovatele systému zpráv IBM MQ jako zvoleného poskytovatele systému zpráv. Aplikace pak může použít objekt továrny JmsFactoryk vytvoření továren připojení a míst určení.

Chcete-li vytvořit továrnu připojení, aplikace volá metodu createConnectionFactory () objektu továrny JmsFactory, jak ukazuje následující příklad:

```
JmsConnectionFactory factory = ff.createConnectionFactory();
```

Tento příkaz vytvoří objekt továrny JmsConnections 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 aby se připojovala ke správci front, který není výchozím správcem front, musí aplikace před vytvořením připojení nastavit příslušné vlastnosti objektu továrny JmsConnection. Informace o tom, jak to provést, viz “Nastavení vlastností objektů IBM MQ classes for JMS” na stránce 203.

Třída továrny JmsFactorytaké obsahuje metody pro vytvoření továren připojení následujících typů:

- JmsQueueConnectionFactory
- JmsTopicConnectionFactory
- Továrna JmsXAConnection
- JmsXAQueueConnectionFactory
- JmsXATopicConnectionFactory

Chcete-li vytvořit objekt fronty, aplikace volá metodu createQueue() objektu továrny JmsFactory, jak ukazuje následující příklad:

```
JmsQueue q1 = ff.createQueue("Q1");
```

Tento příkaz vytvoří objekt JmsQueue s výchozími hodnotami pro všechny jeho vlastnosti. Objekt představuje frontu 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 IBM MQ a volitelně název správce front, který frontu vlastní, 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 fronty viz “Identifikátory URI (Uniform Resource Identifier)” na stránce 215. Alternativní způsob nastavení vlastností objektu JmsQueue naleznete v části “Nastavení vlastností objektů IBM MQ classes for JMS” na stránce 203.

Chcete-li vytvořit objekt Topic, může aplikace použít metodu createTopic() objektu továrny JmsFactory, jak ukazuje následující příklad:

```
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 identifikátor URI tématu jako parametr. Identifikátor URI tématu je řetězec, který určuje 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 “Identifikátory URI (Uniform Resource Identifier)” na stránce 215. Alternativní způsob

nastavení vlastností objektu JmsTopic viz “Nastavení vlastností objektů IBM MQ classes for JMS” na stránce 203.

Poté, co aplikace vytvoří továrnu připojení nebo místo určení, lze tento objekt použít pouze s vybraným poskytovatelem systému zpráv.

Nastavení vlastností objektů IBM MQ classes for JMS

K nastavení vlastností objektů IBM MQ classes for JMS pomocí rozšíření produktu IBM JMS používá aplikace metody rozhraní com.ibm.msg.client.JmsPropertyContext . Podobně k nastavení vlastností objektů IBM MQ classes for Jakarta Messaging pomocí rozšíření produktu IBM JMS používá aplikace metody rozhraní com.ibm.msg.jakarta.client.JmsPropertyContext .

Pro každý datový typ Java obsahuje rozhraní kontextu JmsPropertymetodu pro nastavení hodnoty vlastnosti s tímto datovým typem a metodu pro získání hodnoty vlastnosti s tímto datovým typem. Aplikace například volá metodu setIntProperty () k nastavení vlastnosti s celočíselnou hodnotou a volá metodu getIntProperty () k získání vlastnosti s celočíselnou hodnotou.

Instance tříd v balících com.ibm.mq.jms a com.ibm.mq.jakarta.jms dědí metody odpovídajících rozhraní kontextu JmsProperty. Aplikace proto může tyto metody použít k nastavení vlastností objektů MQConnectionFactory, MQQueue a MQTopic.

Když aplikace vytvoří objekt IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging , všechny vlastnosti s výchozími hodnotami se nastaví automaticky. Když aplikace nastaví vlastnost, nová hodnota nahradí všechny předchozí hodnoty, které tato vlastnost měla. Po nastavení vlastnosti ji nelze odstranit, ale její hodnotu lze změnit.

Pokud se aplikace pokusí nastavit vlastnost na hodnotu, která není platnou hodnotou vlastnosti, IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging vygeneruje výjimku JMSException. Pokud se aplikace pokusí získat vlastnost, která nebyla nastavena, chování je takové, jak je popsáno ve specifikaci JMS . IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging vygenerují výjimku NumberFormatpro primitivní datové typy a vrátí hodnotu null pro odkazované datové typy.

Kromě předdefinovaných vlastností objektu IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging může aplikace nastavit své vlastní vlastnosti. Tyto vlastnosti definované aplikací jsou produktem IBM MQ classes for JMS a IBM MQ classes for Jakarta Messagingignorovány.

Další informace o vlastnostech objektů IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging naleznete v tématu Vlastnosti IBM MQ classes for JMS objektů.

Následující kód je příkladem nastavení vlastností pomocí rozšíření produktu IBM JMS . Kód nastavuje 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, 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 přidružený název aplikace "Moje aplikace".

Poznámka: Správci front spuštění na platformách z/OS nepodporují nastavení názvů aplikací, a toto nastavení je proto ignorováno.

Rozhraní kontextu JmsPropertytaké obsahuje metodu setObjectProperty (), kterou může aplikace použít k nastavení vlastností. Druhý parametr metody je objekt, který zapouzdřuje hodnotu vlastnosti. Následující kód například vytvoří objekt Integer, který zapouzdří celé číslo 1415, a poté zavolá funkci setObjectProperty () pro nastavení vlastnosti PORT továrny připojení na hodnotu 1415:

```
Integer port = new Integer(1415);
factory.setObjectProperty(WMQConstants.WMQ_PORT, port);
```

Tento kód je proto rovnocenný následujícímu prohlášení:

```
factory.setIntProperty(WMQConstants.WMQ_PORT, 1415);
```

Metoda getObjectProperty () naopak vrací objekt, který zapouzdřuje hodnotu vlastnosti.

Implicitní převod hodnoty vlastnosti z jednoho datového typu na jiný

Pokud aplikace používá metodu rozhraní kontextu JmsPropertyk nastavení nebo získání vlastnosti objektu IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging , lze hodnotu vlastnosti implicitně převést z jednoho datového typu na jiný.

Například následující příkaz nastaví vlastnost PRIORITY objektu JmsQueue q1:

```
q1.setStringProperty(WMQConstants.WMQ_PRIORITY, "5");
```

Vlastnost PRIORITY má celočíselnou hodnotu, a proto volání setStringProperty () implicitně převede řetězec "5" (zdrojová hodnota) na celé číslo 5 (cílová hodnota), které se pak stane hodnotou vlastnosti PRIORITY.

Naopak následující příkaz získá vlastnost PRIORITY JmsQueue objektu q1:

```
String s1 = q1.getStringProperty(WMQConstants.WMQ_PRIORITY);
```

Celé číslo 5 (zdrojová hodnota), které je hodnotou vlastnosti PRIORITY, je implicitně převedeno na řetězec "5" (cílová hodnota) voláním funkce getStringProperty ().

Převody podporované produkty IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging jsou uvedeny v souboru Tabulka 34 na stránce 204.

Tabulka 34. Podporované převody z jednoho datového typu na jiný

Datový typ zdroje	**Podporované cílové datové typy**
typ boolean	Řetězec
bajt	int, long, short, String (dlouhý, krátký, řetězec)
ZNAK	Řetězec
dvojitý	Řetězec
float	double, Řetězec
celé číslo	dlouhý, Řetězec
long	Řetězec
short	int, long, Řetězec
Řetězec	boolean, byte, double, float, int, long, short

Obecná pravidla pro podporované převody jsou následující:

- Číselné hodnoty lze převést z jednoho datového typu na jiný za předpokladu, že se během převodu neztratí žádná data. Například hodnotu s datovým typem `int` lze převést na hodnotu s datovým typem `long`, ale nelze ji převést na hodnotu s datovým typem `short`.
- Hodnotu libovolného datového typu lze převést na řetězec.

- Řetězec lze převést na hodnotu libovolného jiného datového typu (kromě `char`). za předpokladu, že řetězec je ve správném formátu pro převod. Pokud se aplikace pokusí převést řetězec, který není ve správném formátu, IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging vygenerují výjimku NumberFormat.
- Pokud se aplikace pokusí o převod, který není podporován, IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging vygenerují výjimku MessageFormatException.

Specifická pravidla pro převod hodnoty z jednoho datového typu na jiný jsou následující:

- Při převodu logické hodnoty na řetězec se hodnota `true` převede 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) je převeden na `false`. Jakýkoli jiný řetězec se převede na `false`.
- Při převodu řetězce na hodnotu s datovým typem `byte`, `int`, `long`nebo `short`musí mít řetězec následující formát:

  [*mezery*] [*znak*] *číslice*

  Význam komponent řetězce je následující:

  mezery
  Volitelné úvodní mezery.

  SIGN
  Volitelné znaménko plus (+) nebo mínus (-).

  číslice
  Souvislá posloupnost číslic (0-9). Musí být uvedena alespoň jedna číslice.

  Po posloupnosti číslic může řetězec obsahovat jiné znaky, které nejsou číslicemi, ale převod se zastaví, jakmile je dosaženo prvního z těchto znaků. Předpokládá se, že řetězec představuje desetinné celé číslo.

  Pokud řetězec není ve správném formátu, IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging vygenerují výjimku NumberFormat.
- Při převodu řetězce na hodnotu s datovým typem `double` nebo `float`musí mít řetězec následující formát:

  [*mezery*] [*znak*] *číslice* [*e_char* [*e_sign*] *e_číslice*]

  Význam komponent řetězce je následující:

  mezery
  Volitelné úvodní mezery.

  SIGN
  Volitelné znaménko plus (+) nebo mínus (-).

  číslice
  Souvislá posloupnost číslic (0-9). Musí být uvedena alespoň jedna číslice.

  e_char
  Exponent, který je buď *E* , nebo *e*.

  e_sign
  Volitelné znaménko plus (+) nebo znaménko minus (-) pro exponent.

  e_číslice
  Souvislá posloupnost číslic (0-9) pro exponent. Pokud řetězec obsahuje exponent, musí být uvedena alespoň jedna číslice.

  Po posloupnosti číslic nebo volitelných znaků představujících exponent může řetězec obsahovat jiné znaky, které nejsou číslicemi, ale převod se zastaví, jakmile je dosaženo prvního z těchto znaků. Předpokládá se, že řetězec představuje desetinné číslo s pohyblivou řádovou čárkou s exponentem, který je mocninou 10.

Pokud řetězec není ve správném formátu, IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging vygenerují výjimku NumberFormat.

- Při převodu číselné hodnoty (včetně hodnoty s datovým typem `byte`). na řetězec se hodnota převede na řetězcovou reprezentaci hodnoty jako desetinné číslo, nikoli na řetězec obsahující pro tuto hodnotu znak ASCII. 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 vlastnosti.

Následující kód například používá metodu setBatchProperties () k nastavení stejných pěti vlastností továrny připojení, jak je uvedeno v části “Nastavení vlastností objektů IBM MQ classes for JMS” na stránce 203. 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 zapouzdřena v rámci objektu nebo reprezentována řetězcem, jak ukazuje příklad.

Metoda setBatchProperties () ověřuje každou vlastnost. Pokud metoda setBatchProperties () nemůže nastavit vlastnost, protože například její hodnota není platná, není nastavena žádná ze zadaných vlastností.

Názvy a hodnoty vlastností

Pokud aplikace používá metody příslušného kontextového rozhraní JmsPropertyk nastavení a získání vlastností objektů IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging , může aplikace zadat názvy a hodnoty vlastností libovolným z následujících způsobů. Každý z doprovodných příkladů ukazuje, jak nastavit vlastnost PRIORITY objektu JmsQueue q1 tak, aby zpráva odeslaná do fronty měla prioritu uvedenou ve volání send ().

Použití názvů a hodnot vlastností, které jsou definovány jako konstanty v rozhraní com.ibm.msg.client.wmq.WMQConstants

Následující příkaz je příkladem, jak zadat názvy a hodnoty vlastností tímto způsobem:

```
q1.setIntProperty(WMQConstants.WMQ_PRIORITY, WMQConstants.WMQ_PRI_APP);
```

Použití názvů a hodnot vlastností, které lze použít v identifikátorech URI (Uniform Resource Identifier) fronty a tématu.

Následující příkaz je příkladem, jak zadat názvy a hodnoty vlastností tímto způsobem:

```
q1.setIntProperty("priority", -2);
```

Tímto způsobem lze určit pouze názvy a hodnoty vlastností cílů.

Použití názvů a hodnot vlastností, které jsou rozpoznány administračním nástrojem produktu IBM MQ JMS .

Následující příkaz je příkladem, jak zadat názvy a hodnoty vlastností tímto způsobem:

```
q1.setStringProperty("PRIORITY", "APP");
```

Krátká forma názvu vlastnosti je také přijatelná, jak je uvedeno v následujícím příkazu:

```
q1.setStringProperty("PRI", "APP");
```

Když aplikace získá vlastnost, vrácená hodnota závisí na způsobu, jakým aplikace určuje název vlastnosti. Pokud například aplikace jako název vlastnosti uvádí konstantu WMQConstants.WMQ_PRIORITY , vrácená hodnota je celé číslo -2:

```
int n1 = getIntProperty(WMQConstants.WMQ_PRIORITY);
```

Stejná hodnota je vrácena, pokud aplikace určuje řetězec "priority" jako název vlastnosti:

```
int n2 = getIntProperty("priority");
```

Pokud však aplikace uvádí řetězec "PRIORITY" nebo "PRI" jako název vlastnosti, vrácená hodnota je řetězec "APP":

```
String s1 = getStringProperty("PRI");
```

Vnitřně IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging ukládají názvy a hodnoty vlastností jako literálové hodnoty definované v odpovídajícím rozhraní WMQConstants. Toto je definovaný kanonický formát pro názvy a hodnoty vlastností. Obecně platí, že pokud aplikace nastaví vlastnosti pomocí jednoho z dalších dvou způsobů zadávání názvů a hodnot vlastností, IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging musí převést názvy a hodnoty ze zadaného vstupního formátu do kanonického formátu. Podobně, pokud aplikace získá vlastnosti pomocí jednoho z dalších dvou způsobů zadávání názvů a hodnot vlastností, IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging musí převést názvy ze zadaného vstupního formátu do kanonického formátu a převést hodnoty z kanonického formátu do požadovaného výstupního formátu. Provedení těchto převodů může mít dopad na výkon.

Názvy a hodnoty vlastností vrácené výjimkami, v trasovacích souborech nebo v protokolu IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging jsou vždy v kanonickém formátu.

Použití rozhraní mapy

Kontextové rozhraní JmsPropertyrozšiřuje rozhraní java.util.Map . Aplikace proto může používat metody rozhraní Map pro přístup k vlastnostem objektu IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging .

Například následující kód vytiskne názvy a hodnoty všech vlastností továrny připojení. Kód používá k získání názvů a hodnot vlastností pouze metody rozhraní Map.

```
// 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 rozhraní mapy nevynechá žádná ověření vlastností ani převody.

Použití rozšíření 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 míst určení za běhu a k nastavení vlastností továren připojení a míst určení.

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 . Následující příklady kódu předpokládají, že tyto balíky byly importovány pomocí následujících příkazů:

```
import com.ibm.jms.*;
import com.ibm.mq.jms.*;
import com.ibm.msg.client.wmq.WMQConstants;
```

Aplikace může použít rozšíření produktu IBM MQ JMS k provedení následujících funkcí:

- Vytvořte továrny připojení a místa určení dynamicky za běhu namísto jejich načítání jako spravovaných objektů z oboru názvů rozhraní JNDI (Java Naming and Directory Interface).
- Nastavení vlastností továren připojení a míst určení

Vytvoření továren připojení

Chcete-li vytvořit továrnu 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 aby se připojovala ke správci front, který není výchozím správcem front, musí aplikace před vytvořením připojení nastavit příslušné vlastnosti objektu MQConnectionFactory . Informace o tom, jak to provést, viz “Nastavení vlastností továren připojení” na stránce 208.

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. Továrna připojení může být buď spravovaný objekt, nebo objekt vytvořený dynamicky za běhu.

Zvažte například následující kód:

```
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, 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.

Aplikace, která používá připojení v reálném čase ke zprostředkovateli, může používat pouze styl publikování/odběru systému zpráv. Nemůže používat dvoubodový styl systému zpráv.

Platné jsou pouze určité kombinace vlastností továrny připojení. Informace o platných kombinacích naleznete v tématu Závislosti mezi vlastnostmi IBM MQ classes for JMS objektů.

Další informace o vlastnostech továrny připojení a o metodách používaných k nastavení jejích vlastností naleznete v tématu Vlastnosti IBM MQ classes for JMS objektů.

Vytváření cílů

Chcete-li vytvořit objekt fronty, může aplikace použít konstruktor MQQueue, jak ukazuje následující příklad:

```
MQQueue q1 = new MQQueue("Q1");
```

Tento příkaz vytvoří objekt MQQueue s výchozími hodnotami pro všechny jeho vlastnosti. Objekt představuje frontu IBM MQ s názvem Q1 , která patří lokálnímu správci front. Tato fronta může být lokální fronta, alias fronta nebo definice vzdálené fronty.

Alternativní forma konstruktoru MQQueue má dva parametry, jak ukazuje následující příklad:

```
MQQueue q2 = new MQQueue("QM2", "Q2");
```

Objekt MQQueue vytvořený tímto příkazem představuje frontu IBM MQ s názvem Q2 , kterou vlastní správce front QM2. Takto identifikovaným správcem front může být lokální správce front nebo vzdálený správce front. Jedná-li se o vzdáleného správce front, musí být produkt IBM MQ konfigurován tak, aby v případě, že aplikace odešle zprávu do tohoto místa určení, mohl 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 IBM MQ a volitelně název správce front, který frontu vlastní, 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 představuje frontu IBM MQ s názvem Q3 , kterou vlastní správce front QM3, a všechny zprávy odeslané do tohoto místa určení jsou trvalé a mají prioritu 5. Další informace o identifikátorech URI fronty viz “Identifikátory URI (Uniform Resource Identifier)” na stránce 215. Alternativní způsob nastavení vlastností objektu MQQueue viz “Nastavení vlastností cílů” na stránce 209.

Chcete-li vytvořit objekt Topic, může aplikace použít konstruktor MQTopic, jak ukazuje následující příklad:

```
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 jednu či 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 “Identifikátory URI (Uniform Resource Identifier)” na stránce 215. Alternativní způsob nastavení vlastností objektu MQTopic viz “Nastavení vlastností cílů” na stránce 209.

Nastavení vlastností cílů

Aplikace může nastavit vlastnosti místa určení voláním příslušných metod místa určení. Cílem může být buď spravovaný objekt, nebo objekt vytvořený dynamicky za běhu.

Zvažte například následující kód:

```
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. Nastavení těchto vlastností má za následek, že všechny zprávy odeslané do místa určení jsou trvalé a mají prioritu 5.

Aplikace může nastavit vlastnosti objektu MQTopic podobným způsobem, jak ukazuje následující příklad:

```
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 přechodné a mají prioritu 0.

Další informace o vlastnostech místa určení a metodách použitých k nastavení jeho vlastností naleznete v tématu Vlastnosti IBM MQ classes for JMS objektů.

Linux AIX *Připojení k produktu IBM MQ z aplikace JMS*

Chcete-li sestavit připojení, aplikace JMS použije objekt **ConnectionFactory** k vytvoření objektu **Connection** a poté spustí připojení.

V případě platformy JMS 2.0 a novější se aplikace obvykle připojují k poskytovateli systému zpráv pomocí objektu **ConnectionFactory** a metody `createContext()` .

Ve starších verzích rozhraní JMS jste nejprve museli použít `createConnection` k vytvoření objektu **Connection** a poté spustit volání připojení `getSession()` k vytvoření objektu **Session** , který by mohl provádět operace systému zpráv.

Objekt **JMSContext** účinně zapouzdřuje objekty **Connection** i **Session** . Chcete-li použít tradiční přístup a vytvořit připojení a objekty relace přímo, viz “Sestavení připojení v aplikaci JMS” na stránce 210 a “Vytvoření relace v aplikaci JMS” na stránce 211.

Chcete-li vytvořit objekt **JMSContext** , aplikace použije metodu `createContext()` objektu **ConnectionFactory** , jak ukazuje následující příklad:

```
ConnectionFactory factory;
Connection connection;
.
.
.
connection = factory.createContext();
```

Při vytvoření připojení JMS vytvoří agent IBM MQ classes for JMS manipulátor připojení (Hconn) a zahájí konverzaci se správcem front.

Poznámka: Všimněte si, že ID procesu aplikace se používá jako výchozí identita uživatele, která má být předána správci front. Pokud je aplikace spuštěna v režimu transportu klienta, musí toto ID procesu existovat na serveru s příslušnými oprávněními. Chcete-li použít jinou identitu, použijte metodu `createConnection(username,password)` .

V 9.3.5 Tento mechanizmus lze také použít k dodání tokenu ověření, viz Získání tokenu ověření od vybraného vydavatele tokenu.

JMS 1.0 *Sestavení připojení v aplikaci JMS*

Chcete-li sestavit připojení v produktu JMS 1.0, aplikace JMS použije objekt ConnectionFactory k vytvoření objektu připojení a poté spustí připojení.

Chcete-li vytvořit objekt připojení, aplikace použije metodu createConnection() objektu ConnectionFactory , jak ukazuje následující příklad:

```
ConnectionFactory factory;
Connection connection;
.
.
.
connection = factory.createConnection();
```

Při vytvoření připojení JMS produkt IBM MQ classes for JMS vytvoří manipulátor připojení (Hconn) a zahájí konverzaci se správcem front.

Rozhraní továrny QueueConnectiona rozhraní továrny TopicConnectionzdědí metodu createConnection() z rozhraní ConnectionFactory . Proto můžete použít metodu createConnection() k vytvoření objektu specifického pro doménu, jak ukazuje následující příklad:

```
QueueConnectionFactory qcf;
Connection connection;
.
.
.
connection = qcf.createConnection();
```

Tento fragment kódu vytváří objekt QueueConnection . Aplikace může nyní na tomto objektu provést operaci nezávislou 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, dojde k výjimce IllegalStates následující 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, 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. Pokud je aplikace spuštěna v režimu transportu klienta, musí toto ID procesu existovat na serveru s příslušnými oprávněními. Chcete-li použít jinou identitu, použijte metodu createConnection(jméno uživatele, heslo).

Specifikace JMS uvádí, že připojení je vytvořeno ve stavu `stopped` . Dokud se nespustí připojení, spotřebitel zpráv, který je přidružen k připojení, nemůže přijímat žádné zprávy. Chcete-li spustit připojení, aplikace použije metodu start () objektu připojení, jak ukazuje následující příklad:

```
connection.start();
```

V 9.3.5 Tento mechanizmus lze také použít k dodání tokenu ověření, viz Získání tokenu ověření od vybraného vydavatele tokenu.

JMS 1.0 *Vytvoření relace v aplikaci JMS*

Chcete-li vytvořit relaci v produktu JMS 1.0, aplikace JMS použije metodu createSession() objektu připojení.

Metoda createSession() má dva parametry:

1. Parametr, který určuje, zda je relace transakční, či nikoli.
2. Parametr, který určuje režim potvrzení pro relaci.

Například následující kód vytvoří relaci, která není transakční a má režim potvrzení AUTO_POTVRZENÍ:

```
Session session;
.
```

```
boolean transacted = false;
session = connection.createSession(transacted, Session.AUTO_ACKNOWLEDGE);
```

Při vytvoření relace JMS vytvoří agent IBM MQ classes for JMS manipulátor připojení (Hconn) a zahájí konverzaci se správcem front.

Objekt relace a z něj vytvořený objekt MessageProducer nebo MessageConsumer nemohou být souběžně používány různými podprocesy aplikace s podporou podprocesů. Nejjednodušším způsobem, jak zajistit, aby tyto objekty nebyly používány souběžně, je vytvořit pro každý podproces samostatný objekt relace.

V 9.3.5 Tento mechanizmus lze také použít k dodání tokenu ověření, viz Získání tokenu ověření od vybraného vydavatele tokenu.

Transakční relace v aplikacích JMS
Aplikace JMS mohou spouštět lokální transakce nejprve vytvořením transakční relace. Aplikace může potvrdit nebo odvolat transakci.

Aplikace JMS mohou spouštět lokální transakce. Lokální transakce je transakce, která zahrnuje pouze změny prostředků správce front, ke kterému je aplikace připojena. Chcete-li spustit lokální transakce, musí aplikace nejprve vytvořit relaci transakcí voláním metody createSession() objektu připojení a určit jako parametr, že relace je transakční. Následně jsou všechny zprávy odeslané a přijaté v rámci relace seskupeny do posloupnosti transakcí. Transakce skončí, když aplikace potvrdí nebo odvolá zprávy, které odeslala a přijala od začátku transakce.

Pro potvrzení transakce aplikace volá metodu commit () objektu Session. Po potvrzení transakce budou všechny zprávy odeslané v rámci transakce k dispozici pro doručení do jiných aplikací a všechny zprávy přijaté v rámci transakce budou potvrzeny tak, aby se server systému zpráv nepokusil o jejich opětovné doručení do aplikace. V doméně typu point-to-point server systému zpráv také odebere přijaté zprávy z jejich front.

Chcete-li odvolat transakci, aplikace volá metodu rollback () objektu Session. Když je transakce odvolána, server systému zpráv zruší všechny zprávy odeslané v rámci transakce a všechny zprávy přijaté v rámci transakce budou znovu k dispozici pro doručení. V doméně typu point-to-point jsou zprávy, které byly přijaty, vloženy zpět do svých front a jsou znovu viditelné pro jiné aplikace.

Nová transakce se spustí automaticky, když aplikace vytvoří relaci transakce nebo zavolá metodu commit () nebo rollback (). Proto má relace s transakcemi vždy aktivní transakci.

Když aplikace zavře relaci transakce, dojde k implicitnímu odvolání transakce. Když aplikace zavře připojení, dojde k implicitnímu odvolání pro všechny relace transakcí připojení.

Pokud aplikace skončí bez zavření připojení, dojde k implicitnímu odvolání také pro všechny relace transakcí připojení.

Transakce je zcela obsažena v relaci s transakcemi. Transakce nemůže zahrnovat relace. To znamená, že není možné, aby aplikace odesílala a přijímala zprávy ve dvou nebo více relacích transakcí a poté všechny tyto akce potvrzovala nebo odvolala jako jedinou transakci.

Režimy potvrzení relací JMS
Každá relace, která není transakční, má režim potvrzení, který určuje, jak jsou potvrzeny zprávy přijaté aplikací. K dispozici jsou tři režimy potvrzení a volba režimu potvrzení ovlivňuje návrh aplikace.

Není-li relace transakční, způsob, jakým jsou zprávy přijaté aplikací potvrzeny, je určen režimem potvrzení relace. Tři režimy potvrzení jsou popsány v následujících odstavcích:

AUTO_POTVRZENÍ

Relace automaticky potvrdí každou zprávu přijatou aplikací.

Pokud jsou zprávy doručeny aplikaci synchronně, relace potvrdí přijetí zprávy při každém úspěšném dokončení volání přijetí. Pokud jsou zprávy doručeny asynchronně, relace potvrdí přijetí zprávy při každém úspěšném dokončení volání metody onMessage() modulu listener pro zprávy.

Pokud aplikace úspěšně obdrží zprávu, ale selhání zabrání výskytu potvrzení, bude zpráva znovu k dispozici pro doručení. Aplikace proto musí být schopna zpracovat zprávu, která je znovu doručena.

POTVRZENÍ_DUPS_OK_

Relace potvrzuje zprávy přijaté aplikací v době, kdy je vybírá.

Použití tohoto režimu potvrzení snižuje množství práce, kterou musí relace provést, ale selhání, které brání potvrzení zprávy, může vést k opětovnému doručení více než jedné zprávy. Aplikace proto musí být schopna zpracovat zprávy, které jsou znovu doručeny.

Omezení: V režimech AUTO_KVITOVAT a DUPS_OK_KVITOVAT produkt JMS nepodporuje aplikaci, která v modulu listener pro zprávy generuje neošetřenou výjimku. To znamená, že zprávy jsou vždy potvrzeny, když se vrátí modul listener pro zprávy, bez ohledu na to, zda byly úspěšně zpracovány (za předpokladu, že jakákoli selhání nejsou závažná a nebrání aplikaci v pokračování). Požadujete-li podrobnější řízení potvrzení zprávy, použijte režim CLIENT_POTVRZENÍ nebo režim transakcí, který poskytuje aplikaci úplnou kontrolu nad funkcemi potvrzení.

POTVRZENÍ_KLIENTA

Aplikace potvrdí přijaté zprávy voláním metody Acknowledge třídy Message.

Aplikace může potvrdit příjem každé zprávy jednotlivě, nebo může přijmout dávku zpráv a volat metodu Acknowledge pouze pro poslední zprávu, kterou obdrží. Když je metoda Acknowledge volána, jsou potvrzeny všechny zprávy přijaté od posledního volání metody.

Ve spojení s kterýmkoli z těchto režimů potvrzení může aplikace zastavit a znovu spustit doručování zpráv v relaci voláním metody Recover třídy Session. Přijaté, ale dříve nepotvrzené zprávy jsou znovu doručeny. Nemusí však být doručeny ve stejném pořadí, v jakém byly doručeny dříve. Mezitím mohly dorazit zprávy s vyšší prioritou a některé původní zprávy mohly vypršet. V doméně typu point-to-point mohly být některé původní zprávy spotřebovány jinou aplikací.

Aplikace může určit, zda je zpráva znovu doručována, kontrolou obsahu pole záhlaví JMSRedelivered zprávy. Aplikace to provede voláním metody getJMSRedelivered() třídy Message.

Vytvoření cílů v aplikaci JMS

Namísto načítání míst určení jako spravovaných objektů z oboru názvů rozhraní JNDI (Java Naming and Directory Interface) může aplikace JMS používat relaci k dynamickému vytváření míst určení za běhu. Aplikace může pomocí identifikátoru URI (Uniform Resource Identifier) identifikovat frontu IBM MQ nebo téma a volitelně zadat jednu či více vlastností objektu Queue nebo Topic.

Použití relace k vytvoření objektů fronty

Chcete-li vytvořit objekt fronty, může aplikace použít metodu createQueue() objektu relace, jak ukazuje následující příklad:

```
Session session;
.
Queue q1 = session.createQueue("Q1");
```

Tento kód vytvoří objekt fronty s výchozími hodnotami pro všechny jeho vlastnosti. Objekt představuje frontu 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 IBM MQ a volitelně také název správce front, který vlastní frontu, a jednu či více vlastností objektu fronty. Následující příkaz obsahuje příklad identifikátoru URI fronty:

```
Queue q2 = session.createQueue("queue://QM2/Q2?persistence=2&priority=5");
```

Objekt fronty vytvořený tímto příkazem představuje 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. Takto identifikovaným správcem front může být lokální správce front nebo vzdálený správce front. Jedná-li se o vzdáleného správce front, musí být produkt IBM MQ konfigurován tak, aby v případě, že aplikace odešle zprávu do tohoto místa určení, mohl produkt WebSphere MQ směrovat zprávu z lokálního správce front

do správce front QM2. Další informace o identifikátorech URI viz “Identifikátory URI (Uniform Resource Identifier)” na stránce 215.

Všimněte si, že parametr v metodě createQueue() obsahuje informace specifické pro poskytovatele. Proto použití metody createQueue() k vytvoření objektu Queue namísto načtení objektu Queue jako spravovaného objektu z oboru názvů JNDI může způsobit, že vaše aplikace bude méně přenosná.

Aplikace může vytvořit objekt TemporaryQueue pomocí metody createTemporaryQueue () objektu Session, jak ukazuje následující příklad:

```
TemporaryQueue q3 = session.createTemporaryQueue();
```

Ačkoli se relace používá k vytvoření dočasné fronty, oborem dočasné fronty je připojení, které bylo použito k vytvoření relace. Jakákoli relace připojení může vytvořit producenty zpráv a spotřebitele zpráv pro dočasnou frontu. Dočasná fronta zůstane až do ukončení připojení nebo aplikace explicitně odstraní dočasnou frontu pomocí metody 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 určuje název modelové fronty, která se používá k vytvoření dynamické fronty, a vlastnost TEMPQPREFIX továrny připojení určuje 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 Session, jak ukazuje následující příklad:

```
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 jednu či více vlastností objektu Téma. 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 přechodné a mají prioritu 0. Další informace o identifikátorech URI tématu viz “Identifikátory URI (Uniform Resource Identifier)” na stránce 215.

Všimněte si, že parametr v metodě createTopic() obsahuje informace specifické pro poskytovatele. Proto použití metody createTopic() k vytvoření objektu Topic namísto načtení objektu Topic jako spravovaného objektu z oboru názvů rozhraní JNDI může způsobit, že vaše aplikace bude méně přenositelná.

Aplikace může vytvořit objekt TemporaryTopic pomocí metody createTemporaryTopic () objektu Session, jak ukazuje následující příklad:

```
TemporaryTopic t3 = session.createTemporaryTopic();
```

Ačkoli se relace používá k vytvoření dočasného tématu, rozsah dočasného tématu je připojení, které bylo použito k vytvoření relace. Jakákoli relace připojení může vytvořit producenty zpráv a spotřebitele zpráv pro dočasné téma. Dočasné téma zůstane v platnosti, dokud připojení neskončí, nebo dokud aplikace explicitně neodstraní dočasné téma pomocí metody TemporaryTopic.delete (), podle toho, co nastane dříve.

Když aplikace vytvoří dočasné téma, produkt IBM MQ classes for JMS vytvoří téma s názvem, který začíná znaky TEMP/*tempTopicPrefix*, kde *tempTopicPrefix* je hodnota vlastnosti TEMPTOPICPREFIX továrny připojení.

Identifikátory URI (Uniform Resource Identifier)

Identifikátor URI fronty je řetězec, který určuje název fronty IBM MQ a volitelně také název správce front, který vlastní frontu, a jednu či více vlastností objektu fronty vytvořeného aplikací. Identifikátor URI tématu je řetězec, který určuje název tématu a volitelně 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 identifikovanou identifikátorem URI.

Správce front může být lokálním správcem front nebo vzdáleným správcem front. Jedná-li se o vzdáleného správce front, musí být produkt IBM MQ konfigurován tak, aby v případě, že aplikace odešle zprávu do fronty, mohl produkt WebSphere MQ směrovat zprávu z lokálního správce front do vzdáleného správce front.

Není-li zadán žádný název, předpokládá se lokální správce front.

qName
Název fronty IBM MQ .

Fronta může být lokální fronta, alias fronta nebo definice vzdálené fronty.

Pravidla pro vytváření názvů front viz Pravidla pro pojmenování IBM MQ objektů.

topicName
Název tématu.

Pravidla pro vytváření názvů témat viz Pravidla pro pojmenování IBM MQ objektů. Vyvarujte se použití zástupných znaků +, #, * a? v názvech témat. Názvy témat obsahující tyto znaky mohou při přihlášení k odběru způsobit neočekávané výsledky. Viz Kombinace řetězců témat.

***propertyName1*, *propertyName2*, ...**
Názvy vlastností objektu Fronta nebo Téma vytvořených aplikací. Tabulka 35 na stránce 216 uvádí platné názvy vlastností, které lze použít v identifikátoru URI.

Nejsou-li zadány žádné vlastnosti, objekt Queue nebo Topic má výchozí hodnoty pro všechny své vlastnosti.

***propertyValue1*, *propertyValue2*, ...**
Hodnoty vlastností objektu Fronta nebo Téma vytvořené aplikací. Tabulka 35 na stránce 216 uvádí platné hodnoty vlastností, které lze použít v identifikátoru URI.

Hranaté závorky ([]) označují volitelnou komponentu a tři tečky (...) znamenají, že seznam dvojic název-hodnota vlastnosti, jsou-li přítomny, může obsahovat jednu nebo více dvojic název-hodnota.

Tabulka 35 na stránce 216 uvádí platné názvy vlastností a platné hodnoty, které lze použít v identifikátorech URI fronty a tématu. Ačkoli nástroj pro administraci produktu IBM MQ JMS používá pro hodnoty vlastností symbolické konstanty, identifikátory URI nemohou obsahovat symbolické konstanty.

Tabulka 35. Názvy vlastností a platné hodnoty pro použití v identifikátorech URI fronty a tématu

Název vlastnosti	**Popis**	**Platné hodnoty**
CCSID	Způsob reprezentace znakových dat v těle zprávy při předávání zprávy produktem IBM MQ classes for JMS do místa určení	• Jakýkoli identifikátor kódované znakové sady podporovaný produktem IBM MQ.
kódování	Způsob reprezentace číselných dat v těle zprávy při předávání zprávy produktem IBM MQ classes for JMS do místa určení	• Libovolná platná hodnota pro pole *Kódování* v deskriptoru zprávy IBM MQ .
Vypršení	Doba platnosti pro zprávy odeslané do cíle	• -2-Jak je uvedeno ve volání send () nebo, není-li uvedeno ve volání send (), výchozí doba platnosti producenta zpráv. • 0-Zpráva odeslaná do místa určení nikdy nevyprší. • Kladné celé číslo určující dobu platnosti v milisekundách.
výběrové vysílání	Nastavení výběrového vysílání pro téma při použití připojení v reálném čase ke zprostředkovateli	Následující seznam obsahuje platné hodnoty. Ke každé hodnotě je přidružena odpovídající hodnota vlastnosti MULTICAST, jak je použita v administračním nástroji produktu IBM MQ JMS . Popis vlastnosti MULTICAST a jejích platných hodnot viz Vlastnosti IBM MQ classes for JMS objektů. • -1-ASCF • 0 - zakázáno • 3-NOTR • 5-SPOLEHLIVÝ • 7-POVOLENO
trvání, perzistence	Perzistence zpráv odeslaných do místa určení	• -2-Jak je uvedeno ve volání send () nebo, není-li uvedeno ve volání send (), výchozí perzistence producenta zpráv. • -1-Jak uvádí atribut *DefPersistence* fronty nebo tématu IBM MQ . • 1-Přechodný. • 2-Trvalé. • 3-Ekvivalentní hodnotě HIGH pro vlastnost PERSISTENCE použité v administračním nástroji IBM MQ JMS . Vysvětlení této hodnoty viz “JMS trvalých zpráv” na stránce 244.

Tabulka 35. Názvy vlastností a platné hodnoty pro použití v identifikátorech URI fronty a tématu (pokračování)		
Název vlastnosti	**Popis**	**Platné hodnoty**
priorita	Priorita zpráv odeslaných do cíle	• -2-Jak je uvedeno ve volání send () nebo, není-li uvedeno ve volání send (), výchozí priorita producenta zpráv. • -1-Jak uvádí atribut *DefPriority* fronty nebo tématu IBM MQ . • Celé číslo v rozsahu 0-9 určující prioritu zpráv odeslaných do místa určení.
targetClient	Zda zprávy odeslané do místa určení obsahují záhlaví MQRFH2	• 0-Zprávy obsahují záhlaví MQRFH2 . • 1-Zprávy neobsahují záhlaví MQRFH2 .

Následující identifikátor URI například identifikuje frontu IBM MQ s názvem Q1 , kterou vlastní lokální správce front. Objekt fronty vytvořený pomocí tohoto identifikátoru URI má výchozí hodnoty pro všechny jeho vlastnosti.

```
queue:///Q1
```

Následující identifikátor URI identifikuje frontu 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 fronty vytvořené 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 přechodné a mají prioritu 0. Zbývající vlastnosti objektu tématu vytvořené pomocí tohoto identifikátoru URI mají své výchozí hodnoty.

```
topic://Sport/Athletics/Results?persistence=1&priority=0
```

Odesílání zpráv v aplikaci JMS

Než bude aplikace JMS moci odesílat zprávy do místa určení, musí nejprve vytvořit objekt MessageProducer pro dané místo určení. Chcete-li odeslat zprávu do místa určení, aplikace vytvoří objekt Message a poté zavolá metodu send () objektu MessageProducer .

Aplikace používá k odesílání zpráv objekt MessageProducer . Aplikace obvykle vytvoří objekt MessageProducer pro specifické místo určení, což může být fronta nebo téma, aby byly všechny zprávy odeslané pomocí producenta zpráv odeslány do stejného místa určení. Proto, aby mohla aplikace vytvořit objekt MessageProducer , musí nejprve vytvořit objekt Queue nebo Topic. Chcete-li získat informace o tom, jak vytvořit objekt Queue nebo Topic, prohlédněte si následující témata:

- “Použití rozhraní JNDI k načtení spravovaných objektů v aplikaci JMS nebo Jakarta Messaging” na stránce 199
- “Použití rozšíření IBM JMS” na stránce 200
- “Použití rozšíření IBM MQ JMS” na stránce 207
- “Vytvoření cílů v aplikaci JMS” na stránce 213

Chcete-li vytvořit objekt MessageProducer , aplikace používá metodu createProducer() objektu Session, jak ukazuje následující příklad:

```
MessageProducer producer = session.createProducer(destination);
```

Parametr `destination` je objekt Queue nebo Topic, který aplikace dříve vytvořila.

Aby mohla aplikace odeslat zprávu, musí vytvořit objekt zprávy. Tělo zprávy obsahuje data aplikace a JMS definuje pět typů těla zprávy:

- Bajty
- Mapa
- Objekt
- Proud
- Text

Každý typ těla zprávy má své vlastní rozhraní JMS , které je podrozhraním rozhraní zprávy, a metodu v rozhraní relace pro vytvoření zprávy s tímto typem těla zprávy. Například rozhraní pro textovou zprávu se nazývá TextMessagea aplikace používá metodu createTextMessage () objektu Session k vytvoření textové zprávy, jak ukazuje následující příkaz:

```
TextMessage outMessage = session.createTextMessage(outString);
```

Další informace o zprávách a tělech zpráv viz “Zprávy produktu JMS” na stránce 139.

K odeslání zprávy používá aplikace metodu send () objektu MessageProducer , jak ukazuje následující příklad:

```
producer.send(outMessage);
```

Aplikace může k odesílání zpráv v obou doménách systému zpráv použít metodu send (). Povaha místa určení určuje, která doména systému zpráv je použita. Avšak TopicPublisher, podrozhraní MessageProducer , které je specifické 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 bez určeného místa určení. V tomto případě musí aplikace při volání metody send () určit místo určení.

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 produkt IBM MQ classes for JMS při odesílání zpráv aplikaci předal zprávu a vrátil řízení zpět aplikaci, aniž by zjistil, zda správce front zprávu přijal bezpečně. To je někdy označováno jako *asynchronní vložení*. Další informace viz “Asynchronní vkládání zpráv do adresáře IBM MQ classes for JMS” na stránce 309.

Příjem zpráv v aplikaci JMS

Aplikace používá spotřebitele zpráv k přijímání zpráv. Trvalý odběratel tématu je spotřebitel zpráv, který přijímá všechny zprávy odeslané do místa určení, včetně zpráv odeslaných v době, kdy je odběratel neaktivní. Aplikace může vybrat, které zprávy chce přijmout pomocí selektoru zpráv, a může přijímat zprávy asynchronně pomocí modulu listener pro zprávy.

Aplikace používá k přijímání zpráv objekt MessageConsumer . Aplikace vytvoří objekt MessageConsumer pro specifické místo určení, což může být fronta nebo téma, takže všechny zprávy přijaté pomocí spotřebitele zpráv budou přijaty ze stejného místa určení. Aby mohla aplikace vytvořit objekt MessageConsumer , musí nejprve vytvořit objekt Queue nebo Topic. Chcete-li získat informace o tom, jak vytvořit objekt Queue nebo Topic, prohlédněte si následující témata:

- “Použití rozhraní JNDI k načtení spravovaných objektů v aplikaci JMS nebo Jakarta Messaging” na stránce 199
- “Použití rozšíření IBM JMS” na stránce 200

- “Použití rozšíření IBM MQ JMS” na stránce 207
- “Vytvoření cílů v aplikaci JMS” na stránce 213

Chcete-li vytvořit objekt MessageConsumer , aplikace použije metodu createConsumer() objektu Session, jak ukazuje následující příklad:

```
MessageConsumer consumer = session.createConsumer(destination);
```

Parametr `destination` je objekt Queue nebo Topic, který aplikace dříve vytvořila.

Aplikace poté použije metodu receive () objektu MessageConsumer k přijetí zprávy z místa určení, jak ukazuje následující příklad:

```
Message inMessage = consumer.receive(1000);
```

Parametr volání receive () určuje, jak dlouho v milisekundách metoda čeká na doručení vhodné zprávy, pokud není žádná zpráva k dispozici okamžitě. Pokud tento parametr vynecháte, volání se zablokuje na dobu neurčitou, dokud nepřijde vhodná zpráva. Pokud nechcete, aby aplikace čekala na zprávu, použijte místo toho metodu receiveNoWait ().

Metoda receive () vrací zprávu určitého typu. Když například aplikace obdrží textovou zprávu, objekt vrácený voláním receive () je objekt TextMessage .

Deklarovaný typ objektu vrácený voláním receive () je však objekt Message. Proto, aby bylo možné extrahovat data z těla zprávy, která byla právě přijata, musí být aplikace přetypován ze třídy zprávy na specifičtější podtřídu, například TextMessage. Není-li typ zprávy znám, může aplikace k určení typu použít operátor `instanceof` . Vždy je dobrým zvykem, aby aplikace před přetypováním určila typ zprávy, 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 obdrží zprávy z místa určení, které je konfigurováno pro dopředné čtení, všechny přechodné zprávy, které jsou ve vyrovnávací paměti dopředného čtení při ukončení aplikace, budou zrušeny.

V doméně publikování/odběru produkt JMS identifikuje dva typy spotřebitele zpráv, odběratele přechodných témat a odběratele trvalých témat, které jsou popsány v následujících dvou sekcích.

Přechodní odběratelé témat

Přechodný odběratel témat přijímá pouze ty zprávy, které jsou publikovány v době, kdy je odběratel aktivní. Přechodný odběr se spustí, když aplikace vytvoří odběratele netrvalých témat a skončí, když aplikace zavře odběratele nebo když odběratel spadne mimo rozsah. Jako rozšíření v produktu IBM MQ classes for JMSobdrží přechodný odběratel témat také zachovaná publikování.

Chcete-li vytvořit odběratele netrvalých témat, může aplikace použít metodu createConsumer() nezávislou na doméně a určit objekt Topic jako místo určení. Alternativně může aplikace použít metodu createSubscriber() specifickou pro doménu, jak ukazuje následující příklad:

```
TopicSubscriber subscriber = session.createSubscriber(topic);
```

Parametr `topic` je objekt tématu, který aplikace vytvořila již dříve.

Odběratelé trvalých témat

Omezení: Aplikace nemůže vytvořit trvalé odběratele témat při použití připojení v reálném čase ke zprostředkovateli.

Trvalý odběratel témat obdrží všechny zprávy, které jsou publikovány během životnosti trvalého odběru. Tyto zprávy zahrnují všechny zprávy, které jsou publikovány, když není odběratel aktivní. Jako rozšíření v produktu IBM MQ classes for JMSodběratel trvalého tématu také přijímá zachovaná publikování.

Chcete-li vytvořit trvalého odběratele tématu, aplikace použije metodu createDurableSubscriber () objektu Session, jak ukazuje následující příklad:

```
TopicSubscriber subscriber = session.createDurableSubscriber(topic, "D_SUB_000001");
```

Při volání funkce createDurableSubscriber () je prvním parametrem objekt tématu, který aplikace dříve vytvořila, a druhým parametrem je název, který se používá k identifikaci trvalého odběru.

Relace použitá k vytvoření trvalého odběratele 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 určit nastavením vlastnosti CLIENTID objektu ConnectionFactory . Alternativně může aplikace zadat identifikátor klienta voláním metody setClientID () objektu 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 pokračovat v používání trvalého odběru, který byl vytvořen dříve, musí aplikace vytvořit trvalého odběratele tématu s použitím relace se stejným identifikátorem klienta, který je přidružen k trvalému odběru, a se stejným názvem odběru.

Trvalý odběr se spustí, když aplikace vytvoří trvalého odběratele tématu s použitím identifikátoru klienta a názvu odběru, pro který v současné době neexistuje žádný trvalý odběr. Trvalý odběr se však neukončí, když aplikace zavře trvalého odběratele tématu. Chcete-li ukončit trvalý odběr, musí aplikace volat metodu unsubscribe () objektu Session, který má stejný identifikátor klienta jako ten, který je přidružen k trvalému odběru. Parametr ve volání unsubscribe () je název odběru, jak ukazuje následující příklad:

```
session.unsubscribe("D_SUB_000001");
```

Oborem trvalého odběru je správce front. Pokud v jednom správci front existuje trvalý odběr a aplikace připojená k jinému správci front vytvoří trvalý odběr se stejným identifikátorem klienta a názvem odběru, jsou oba trvalé odběry zcela nezávislé.

Selektory zpráv

Aplikace může určit, že následující volání receive () vrátí pouze ty zprávy, které splňují určitá kritéria. Při vytváření objektu MessageConsumer může aplikace určit výraz SQL (Structured Query Language), který určuje, které zprávy se načítají. Tento výraz SQL se nazývá *selektor zpráv*. Selektor zpráv může obsahovat názvy polí záhlaví zprávy JMS a vlastnosti zprávy. Informace o tom, jak sestavit selektor zpráv, viz “Selektory zpráv v adresáři JMS” na stránce 140.

Následující příklad ukazuje, jak může aplikace vybrat zprávy na základě uživatelem definované vlastnosti s názvem myProp:

```
MessageConsumer consumer;
.
consumer = session.createConsumer(destination, "myProp = 'blue'");
```

Specifikace JMS neumožňuje aplikaci změnit selektor zpráv 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, musí vytvořit spotřebitele zpráv pro každý selektor zpráv.

Všimněte si, že když je aplikace připojena ke správci front verze 7, nemá vlastnost MSGSELECTION továrny připojení žádný vliv. Chcete-li optimalizovat výkon, je veškerý výběr zpráv proveden správcem front.

Potlačení lokálních publikací

Aplikace může vytvořit spotřebitele zpráv, který ignoruje publikování publikovaná ve vlastním připojení spotřebitele. Aplikace to provede nastavením třetího parametru ve volání createConsumer() na `true`, jak ukazuje následující příklad:

```
MessageConsumer consumer = session.createConsumer(topic, null, true);
```

Při volání createDurableSubscriber () to aplikace provede nastavením čtvrtého parametru na hodnotu `true`, jak ukazuje následující příklad.

```
String selector = "company = 'IBM'";
TopicSubscriber subscriber = session.createDurableSubscriber(topic, "D_SUB_000001",
                                                              selector, true);
```

Asynchronní doručování zpráv

Aplikace může přijímat zprávy asynchronně registrací modulu listener pro zprávy u spotřebitele zpráv. Modul listener pro zprávy má metodu s názvem onMessage, která se volá asynchronně, když je k dispozici vhodná zpráva a jejímž účelem je zpracovat zprávu. Mechanismus je znázorněn následujícím kódem:

V 9.3.0 JM 3.0 V 9.3.0

```
import jakarta.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
```

JMS 2.0

```
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 příjem zpráv asynchronně pomocí modulů listener pro zprávy, ale ne pro obojí. Pokud aplikace potřebuje přijímat zprávy synchronně a asynchronně, musí vytvořit oddělené relace.

Jakmile je relace nastavena tak, aby přijímala zprávy asynchronně, nelze pro tuto relaci nebo pro objekty vytvořené z této relace volat následující metody:

- MessageConsumer.receive ()
- MessageConsumer.receive (long)
- MessageConsumer.receiveNoWait ()
- Session.acknowledge()
- MessageProducer.send (Destination, Message)
- MessageProducer.send (Destination, Message, int, int, long).
- MessageProducer.send (Message)
- MessageProducer.send (Message, int, int, long)
- MessageProducer.send (Destination, Message, CompletionListener).
- MessageProducer.send (Destination, Message, 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, řetězec, logická hodnota)
- Session.createDurableSubscriber(téma, řetězec)
- Session.createDurableSubscriber(téma, řetězec, řetězec, logická hodnota)
- Session.createMapMessage()
- Session.createMessage()
- Session.createObjectMessage()
- Session.createObjectMessage(serializable)
- Session.createProducer(místo určení)
- Session.createQueue(řetězec)
- Session.createStreamMessage()

- Session.createTemporaryQueue()
- Session.createTemporaryTopic()
- Session.createTextMessage()
- Session.createTextMessage(řetězec)
- Session.createTopic()
- Session.getAcknowledgeMode()
- Session.getMessageListener()
- Session.getTransacted()
- Session.rollback()
- Session.unsubscribe(Řetězec)

Je-li volána některá z těchto metod, dojde k výjimce JMSException obsahující zprávu:

```
JMSCC0033: Volání synchronní metody není povoleno, když se relace používá asynchronně: 'název metody'
```

je vyvolána.

Příjem nezpracovatelných zpráv

Aplikace může přijmout zprávu, kterou nelze zpracovat. Může existovat několik příčin, proč zprávu nelze zpracovat, například zpráva může mít nesprávný formát. Takové zprávy jsou popsány jako nezpracovatelné zprávy a vyžadují speciální zacházení, aby se zabránilo rekurzivnímu zpracování zprávy.

Podrobnosti o tom, jak zpracovat nezpracovatelné zprávy, viz “Zpracování nezpracovatelných zpráv v adresáři IBM MQ classes for JMS” na stránce 225.

Přizpůsobení velikostí vyrovnávací paměti tak, aby vyhovovaly přijatým zprávám

Když aplikace, která není součástí systémuJMS , přijme zprávu z produktu IBM MQ , musí aplikace poskytnout vyrovnávací paměť pro zprávy, do které se má zpráva zapsat. Aplikace JMS nemusí ručně vytvářet vyrovnávací paměť. Produkt IBM MQ classes for JMS automaticky vytvoří a nastaví velikost vyrovnávacích pamětí zpráv tak, aby vyhovovaly velikosti přijatých zpráv. Pro většinu aplikací poskytují automaticky spravované vyrovnávací paměti vhodnou rovnováhu výkonu a pohodlí pro vývojáře aplikací. Za určitých okolností může být výhodné zadat počáteční velikost vyrovnávací paměti zpráv ručně. Výchozí počáteční velikost vyrovnávací paměti pro příjem IBM MQ JMS je 4 kB. Pokud aplikace bude vždy přijímat zprávy o velikosti 256 kB, může být vhodnější konfigurovat počáteční velikost vyrovnávací paměti na 256 kB. To může zabránit tomu, aby se agent IBM MQ classes for JMS pokusil přijmout zprávu do vyrovnávací paměti o velikosti 4 kB a nepodařilo se jí ji přijmout, než ji změní na 256 kB a úspěšně ji obdrží. V případě aplikací připojených ke klientovi se může vyhnout potřebě potenciálně zbytečného zpětného přenosu sítě, zatímco IBM MQ classes for JMS určí správnou velikost vyrovnávací paměti, která se má použít.

Počáteční velikost vyrovnávací paměti lze konfigurovat nastavením vlastnosti com.ibm.mq.jmqi.defaultMaxMsgSize Java na zvolenou hodnotu v bajtech. Všimněte si, že tato vlastnost ovlivňuje všechny aplikace produktu IBM MQ JMS , které jsou spuštěny v rámci produktu Java Virtual Machine, takže dbejte na to, abyste nepříznivě neovlivňovaly ostatní spotřebitele zpráv, kteří přijímají zprávy jiné velikosti.

Produkt IBM MQ classes for JMS se stále pokouší automaticky zmenšit velikost vyrovnávací paměti, pokud je přijato několik zpráv menších, než je nakonfigurovaná velikost. Standardně k tomu dochází, když je přijato 10 zpráv, které jsou všechny menší než velikost vyrovnávací paměti. Je-li například v řádku přijato 10 zpráv o velikosti 128 kB, bude velikost vyrovnávací paměti snížena na hodnotu z 256 kB na 128 kB. Poté se znovu zvýší, když jsou přijímány větší zprávy. Je možné konfigurovat počet zpráv, které musí být přijaty před zmenšením velikosti vyrovnávací paměti. To může být užitečné například v případě, že je známo, že aplikace přijímá pět velkých zpráv následovaných 10 menšími zprávami a poté dalších pět velkých zpráv. Při výchozím nastavení by vyrovnávací paměť byla snížena po přijetí 10 menších zpráv a pro větší zprávy by bylo nutné ji znovu zvýšit. Systémovou vlastnost Java com.ibm.mq.jmqi.smallMsgBufferReductionThreshold lze nastavit na počet zpráv, které musí být přijaty

před zmenšením velikosti vyrovnávací paměti. V tomto příkladu by mohla být nastavena na 20, aby se zabránilo 10 menším zprávám zmenšit velikost vyrovnávací paměti.

Vlastnosti lze nastavit nezávisle na sobě. Můžete se například rozhodnout ponechat počáteční velikost vyrovnávací paměti na výchozí hodnotě 4 kB, ale zvýšit hodnotu com.ibm.mq.jmqi.smallMsgBufferReductionThreshold , takže po zvýšení velikosti vyrovnávací paměti zůstane tato velikost delší.

Pokud se pro aplikace JMS v záznamech statistiky MQI zobrazí velký počet návratových kódů MQRC_TRUNCATED_MSG_FAILED (2080), může to znamenat, že byste měli prospěch z konfigurace vyšší počáteční velikosti vyrovnávací paměti pro tyto aplikace, nebo snížení frekvence, s jakou se zmenšují velikosti vyrovnávací paměti. Je však důležité si uvědomit, že u dlouho běžící aplikace pravděpodobně uvidíte pouze velmi malý počet návratových kódů MQRC_TRUNCATED_MSG_FAILED. Důvodem je, že vyrovnávací paměť je obvykle zvětšena na správnou velikost ihned po přijetí první velké zprávy a není zmenšena, pokud není přijat počet menších zpráv. Je proto možné, že velký počet MQRC_TRUNCATED_MSG_FAILED označuje jiné špatné postupy aplikace, jako je připojení k produktu IBM MQ , aby před odpojením obdržel pouze jednu nebo dvě zprávy.

Načtení uživatelských dat předplatného

Pokud jsou zprávy, které aplikace IBM MQ classes for JMS spotřebovává z fronty, vloženy administrativně definovaným trvalým odběrem, musí aplikace přistupovat k informacím o uživatelských datech, které jsou přidruženy k odběru. Tyto informace jsou přidány do zprávy jako vlastnost.

Když je zpráva spotřebována z fronty, která obsahuje záhlaví RFH2 se složkou MQPS, hodnota, která je přidružena ke klíči Sud, pokud existuje, je přidána jako řetězcová vlastnost do objektu JMS Message vráceného do aplikace IBM MQ classes for JMS . Chcete-li povolit načtení této vlastnosti ze zprávy, konstantu JMS_IBM_SUBSCRIPTION_USER_DATA v rozhraní JmsConstants lze použít s následující metodou k získání uživatelských dat odběru:

- V 9.3.0 JM 3.0 V 9.3.0 jakarta.jms.Message.getStringProperty(java.lang.String)
- JMS 2.0 javax.jms.Message.getStringProperty(java.lang.String)

V následujícím příkladu je administrativní trvalý odběr definován pomocí 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 publikovaných do řetězce tématu PUBLIC jsou vloženy do fronty MY.SUBSCRIPTION.Q. Uživatelská data 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:

V 9.3.0 JM 3.0 V 9.3.0
```
jakarta.jms.Message.getStringProperty(JmsConstants.JMS_IBM_SUBSCRIPTION_USER_DATA);
```

JMS 2.0
```
javax.jms.Message.getStringProperty(JmsConstants.JMS_IBM_SUBSCRIPTION_USER_DATA);
```

Poté se vrátí následující řetězec:

```
Administrative durable subscription to put message to the queue MY.SUBSCRIPTION.Q
```

Související pojmy

“Záhlaví MQRFH2 a JMS” na stránce 145

Související úlohy

Definování administrativního odběru

Související odkazy

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 uvolnění prostředků. Nepovolit ukončení aplikace s aktivním komprimovaným trasováním.

Samotné uvolňování paměti nemůže včas uvolnit všechny prostředky IBM MQ classes for JMS a IBM MQ , zvláště pokud aplikace vytvoří mnoho objektů JMS s krátkou životností na úrovni relace nebo nižší. Proto je důležité, aby aplikace zavřeli objekt Connection, Session, MessageConsumernebo MessageProducer , když již není potřeba.

Pokud aplikace skončí bez zavření připojení, dojde k implicitnímu odvolání pro všechny relace transakcí připojení. Chcete-li zajistit potvrzení všech změn provedených aplikací, zavřete připojení explicitně před uzavřením aplikace.

K zavření objektů JMS nepoužívejte v aplikaci finalizéry. Protože finalizéry nemusí být volány, prostředky nemusí být uvolněny. Když je připojení uzavřeno, zavře všechny relace, které z něj byly vytvořeny. Podobně se při zavření relace zavřou MessageConsumers a MessageProducers vytvořené z relace. Zvažte však uzavření relací, MessageConsumersa MessageProducers explicitně, abyste zajistili včasné uvolnění prostředků.

Je-li aktivována komprese trasování, funkce System.Halt() se vypne a nekontrolovaná ukončení prostředí JVM budou pravděpodobně mít za následek poškození trasovacího souboru. Pokud je to možné, vypněte prostředek trasování, když jste shromáždili informace o trasování, které potřebujete. Pokud trasujete aplikaci až do nestandardního ukončení, použijte nekomprimovaný výstup trasování.

Poznámka: Chcete-li se odpojit od správce front, aplikace JMS vyvolá metodu close () na objektu připojení.

Zpracování nezpracovatelných zpráv v adresáři IBM MQ classes for JMS

Nezpracovatelnou zprávou je zpráva, kterou přijímající aplikace nemůže zpracovat. Je-li aplikace doručena nezpracovatelnou zprávu a byla-li několikrát odvolána, produkt IBM MQ classes for JMS ji může přesunout do fronty vrácení.

Nezpracovatelnou zprávou je zpráva, kterou přijímající aplikace nemůže zpracovat. Zpráva může mít neočekávaný typ nebo může obsahovat informace, které nelze zpracovat logikou aplikace. Pokud je aplikace doručena nezpracovatelnou zprávu, aplikace ji nebude moci zpracovat a vrátí ji zpět do fronty, ze které pochází. Standardně produkt IBM MQ classes for JMS opakovaně doručí zprávu do aplikace. To může vést k tomu, že aplikace uvízne ve smyčce, která se neustále pokouší zpracovat nezpracovatelnou zprávu a vrátit ji zpět.

Aby se tomu zabránilo, může produkt IBM MQ classes for JMS detekovat nezpracovatelné zprávy a přesunout je do alternativního místa určení. Chcete-li to provést, produkt IBM MQ classes for JMS použije následující vlastnosti:

- Hodnota pole BackoutCount v deskriptoru MQMD zprávy, která byla zjištěna.
- IBM MQ Atributy fronty **BOTHRESH** (prahová hodnota vrácení) a **BOQNAME** (fronta vrácení) pro vstupní frontu obsahující zprávu.

Kdykoli aplikace odvolá zprávu, správce front automaticky zvýší hodnotu pole BackoutCount pro zprávu.

Když produkt IBM MQ classes for JMS zjistí zprávu, která má hodnotu BackoutCount větší než nula, porovnají hodnotu BackoutCount s hodnotou atributu **BOTHRESH** .

- Je-li hodnota BackoutCount menší než hodnota atributu **BOTHRESH** , IBM MQ classes for JMS ji doručí aplikaci ke zpracování.
- Je-li však hodnota BackoutCount větší nebo rovna hodnotě **BOTHRESH**, je zpráva považována za nezpracovatelnou zprávu. V této situaci IBM MQ classes for JMS pak přesune zprávu do fronty uvedené atributem **BOQNAME** . Pokud zprávu nelze vložit do fronty vrácení, bude buď přesunuta do fronty nedoručených zpráv správce front, nebo vyřazena, v závislosti na volbách sestavy zprávy.

Poznámka:

- Pokud je atribut **BOTHRESH** ponechán na výchozí hodnotě 0, je zpracování nezpracovatelných zpráv zakázáno. To znamená, že všechny nezpracovatelné zprávy jsou vloženy zpět do vstupní fronty.
- Další věc, kterou je třeba poznamenat, je, že IBM MQ classes for JMS dotazujte atributy **BOTHRESH** a **BOQNAME** pro frontu při prvním zjištění zprávy, která má hodnotu BackoutCount větší než nula. Hodnoty těchto atributů se pak uloží do mezipaměti a použijí se vždy, když produkt IBM MQ classes for JMS narazí na zprávu, která má hodnotu 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 se jedná o základní lokální frontu. To je důležité v případě, že aplikace systému JMS spotřebovává zprávy z alias front nebo z front klastru.
- Pro systém zpráv publikování/odběru se vytvoří spravovaná fronta, která bude uchovávat zprávy pro aplikaci. Dotaz IBM MQ classes for JMS na spravovanou frontu určuje hodnoty atributů **BOTHRESH** a **BOQNAME** .

  Spravovaná fronta je vytvořena z modelové fronty přidružené k objektu Topic, k jehož odběru se aplikace přihlásí, a dědí hodnoty atributů **BOTHRESH** a **BOQNAME** z modelové fronty. Použitá modelová fronta závisí na tom, zda přijímající aplikace vzala trvalý nebo netrvalý odběr:

  – Modelová fronta použitá pro trvalé odběry je určena atributem **MDURMDL** tématu. Výchozí hodnota tohoto atributu je SYSTEM.DURABLE.MODEL.QUEUE.

  – Pro dočasné odběry je použitá modelová fronta určena atributem **MNDURMDL** . Výchozí hodnota atributu **MNDURMDL** je SYSTEM.NDURABLE.MODEL.QUEUE.

Při dotazování na atributy **BOTHRESH** a **BOQNAME** se jedná o IBM MQ classes for JMS:

- Otevřete lokální frontu nebo cílovou frontu pro alias frontu.
- Informujte se o atributech **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 fronty, závisí na použité verzi IBM MQ classes for JMS :

- V případě systému IBM MQ classes for JMS v systému IBM MQ 9.1.0 Fix Pack 1 a starším, nebo v případě systému IBM MQ 9.1.1, je-li lokální fronta nebo cílová fronta pro alias fronty frontou klastru, pak IBM MQ classes for JMS otevře 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 přístup k lokální instanci fronty klastru.

  IBM MQ classes for JMS otevřete všechny ostatní typy lokální fronty s volbami otevření MQOO_INQUIRE a MQOO_FAIL_IF_QUIESCING. Aby mohl uživatel IBM MQ classes for JMS dotazovat hodnoty atributů, musí mít uživatel, který spustil přijímající aplikaci, přístup s dotazem na lokální frontu.
- Při použití konzoly IBM MQ classes for JMS v systému IBM MQ 9.1.0 Fix Pack 2 a novějším nebo v systému IBM MQ 9.1.2 a novějším musí mít uživatel, který spustil přijímající aplikaci, přístup s dotazem na lokální frontu, bez ohledu na typ fronty.

Chcete-li přesunout nezpracovatelné zprávy buď do fronty vrácení, nebo do fronty nedoručených zpráv správce front, musíte udělit uživateli, který spustil aplikaci, oprávnění put a passall .

Zpracování nezpracovatelných zpráv pro synchronní aplikace

Pokud aplikace přijímá zprávy synchronně voláním jedné z následujících metod, systém IBM MQ classes for JMS v rámci transakce, která byla aktivní, když se aplikace pokusila zprávu získat, zprávu znovu zobrazí:

- 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 (dlouhý časový limit)
- MessageConsumer.receiveNoWait ()
- QueueReceiver.receive ()
- QueueReceiver.receive (dlouhý časový limit)
- QueueReceiver.receiveNoWait ()
- TopicSubscriber.receive ()
- TopicSubscriber.receive (dlouhý časový limit)
- TopicSubscriber.receiveNoWait ()

To znamená, že pokud aplikace používá buď transakční kontext JMS , nebo relaci, nebude přesun zprávy do fronty vrácení potvrzen, dokud nebude transakce potvrzena.

Je-li atribut **BOTHRESH** nastaven na jinou hodnotu než na nulu, měl by být také nastaven atribut **BOQNAME** . Pokud je parametr **BOTHRESH** nastaven na hodnotu větší než nula a hodnota **BOQNAME** nebyla nastavena, je chování určeno volbami sestavy zprávy:

- Pokud má zpráva nastavenou volbu sestavy MQRO_DISCARD_MSG , zpráva se zruší.
- Pokud je pro zprávu zadána volba sestavy MQRO_DEAD_LETTER_Q , pokusí se konzola 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 , produkt IBM MQ classes for JMS se pokusí vložit zprávu do fronty nedoručených zpráv pro správce front.

V případě, že pokus o vložení zprávy do fronty nedoručených zpráv z nějakého důvodu selže, je to, co se stane se zprávou, určeno tím, zda přijímající aplikace používá kontext nebo relaci JMS s transakčním nebo netransakčním:

- Pokud přijímající aplikace používá transakční kontext JMS nebo relaci a transakce je potvrzena, zpráva se zruší.
- Pokud přijímající aplikace používá transakční kontext JMS nebo relaci a odvolá transakci, zpráva se vrátí do vstupní fronty.
- Pokud přijímající aplikace vytvořila netransakční JMS kontext nebo relaci, zpráva se zruší.

Zpracování nezpracovatelných zpráv pro asynchronní aplikace

Pokud aplikace přijímá zprávy asynchronně prostřednictvím modulu MessageListener, nezpracovatelné zprávy systému IBM MQ classes for JMS bez ovlivnění doručení zpráv. Proces zařazení do fronty probíhá mimo jakoukoli pracovní jednotku přidruženou ke skutečnému doručení zprávy do aplikace.

Pokud je parametr **BOTHRESH** nastaven na hodnotu větší než nula a hodnota **BOQNAME** nebyla nastavena, je chování určeno volbami sestavy zprávy:

- Pokud má zpráva nastavenou volbu sestavy MQRO_DISCARD_MSG , zpráva se zruší.
- Pokud je pro zprávu zadána volba sestavy MQRO_DEAD_LETTER_Q , pokusí se konzola 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 , produkt IBM MQ classes for JMS se pokusí vložit zprávu do fronty nedoručených zpráv pro správce front.

Pokud pokus o vložení zprávy do fronty nedoručených zpráv z nějakého důvodu selže, IBM MQ classes for JMS vrátí zprávu do vstupní fronty.

Informace o způsobu, jakým specifikace aktivace a ConnectionConsumers zpracovávají nezpracovatelné zprávy, naleznete v tématu Odebrání zpráv z fronty v ASF.

Co se stane se zprávou, když je přesunuta do fronty vrácení

Když je nezpracovatelná zpráva znovu zařazena do fronty pro vrácení, IBM MQ classes for JMS přidá do ní záhlaví RFH2 (pokud ještě žádné nebylo) a aktualizuje některá pole v deskriptoru zprávy (MQMD).

Pokud nezpracovatelná zpráva obsahuje záhlaví RFH2 (například proto, že se jednalo o zprávu JMS), změní agent IBM MQ classes for JMS při přesouvání zprávy do fronty vrácení do následujících polí v deskriptoru MQMD:

- Pole BackoutCount je resetováno na nulu.
- Pole vypršení platnosti zprávy je aktualizováno tak, aby odráželo zbývající vypršení platnosti v době, kdy aplikace JMS přijala nezpracovatelnou zprávu.

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, kdy aplikace JMS přijala nezpracovatelnou zprávu.
- Pole Formát zprávy se změní na MQHRF2.
- Pole CCSID se změní na 1208.
- Pole kódování je upraveno na hodnotu 273.

Kromě toho jsou pole CCSID a Kódování ze nezpracovatelné zprávy zkopírována do polí CCSID a Kódování v záhlaví RFH2 , aby se zajistilo, že řetězení záhlaví zprávy ve frontě zpětného volání je správné.

Související pojmy

“Zpracování nezpracovatelných zpráv v ASF” na stránce 326
V rámci zařízení aplikačního serveru je zpracování nezpracovatelných zpráv ošetřeno mírně jinak než jinde v produktu IBM MQ classes for JMS.

Výjimky v souboru IBM MQ classes for JMS

Aplikace IBM MQ classes for JMS musí zpracovat výjimky, které jsou vyvolány voláními rozhraní API produktu JMS nebo které jsou doručeny obslužné rutině výjimek.

Produkt IBM MQ classes for JMS hlásí běhové problémy vyvoláním výjimek. Typ výjimek, které jsou vyvolány, a způsob, jakým musí být tyto výjimky zpracovány, závisí na verzi specifikace JMS, kterou používá vaše aplikace:

- Metody na rozhraních, která jsou definována v rozhraní JMS 1.1 a dříve vyvolala kontrolované výjimky. Základní třída těchto výjimek je `JMSException`. Další informace o tom, jak zpracovat kontrolované výjimky, viz “Zpracování kontrolovaných výjimek” na stránce 229.
- Metody na rozhraních přidaných v produktu JMS 2.0 generují nekontrolované výjimky. Základní třída pro tyto výjimky je `JMSRuntimeException`. Další informace o tom, jak zpracovat nekontrolované výjimky, viz “Zpracování nekontrolovaných výjimek” na stránce 232.

Můžete také zaregistrovat `ExceptionListener` s JMS `Connection` nebo `JMSContext`. Třídy produktu MQ pro platformu JMS poté upozorní produkt `ExceptionListener` , pokud je zjištěn problém s připojením ke správci front nebo pokud dojde k problému při asynchronním pokusu o doručení zprávy. Další informace viz “ExceptionListeners” na stránce 235.

Související pojmy

IBM MQ třídy pro JMS

Související odkazy

Výjimka ASYNCEXCEPTION

Zpracování kontrolovaných výjimek
Metody na rozhraních, která jsou definována v rozhraní JMS 1.1 nebo dříve, generují kontrolované výjimky. Základní třída pro tyto výjimky je `JMSException`. Proto zachycení `JMSExceptions` poskytuje generický způsob zacházení s těmito typy výjimek.

Každý `JMSException` zapouzdřuje následující informace:

- Zpráva výjimky specifická pro poskytovatele, kterou může aplikace získat voláním metody `Throwable.getMessage()`.
- Kód chyby specifický pro poskytovatele, který může aplikace získat voláním metody `JMSException.getErrorCode()`.
- Připojená výjimka. Výjimka vyvolaná voláním rozhraní API JMS 1.1 je často výsledkem problému nižší úrovně, který je ohlášen jinou výjimkou, která je propojena s touto výjimkou. Vaše aplikace může získat připojenou výjimku voláním metody `JMSException.getLinkedException()` nebo metody `Throwable.getCause()`.

Používáte-li rozhraní API platformy JMS 1.1 , většina výjimek, které jsou vyvolány produktem IBM MQ classes for JMS , jsou instance podtříd produktu `JMSException`. Tyto podtřídy implementují rozhraní `com.ibm.msg.client.jms.JmsExceptionDetail` , které poskytuje následující další informace:

- Vysvětlení zprávy výjimky. Aplikace může tuto zprávu získat voláním metody `JmsExceptionDetail.getExplanation()`.
- Doporučená odezva uživatele na výjimku. Aplikace může tuto zprávu získat voláním metody `JmsExceptionDetail.getUserAction()`.
- Klíče pro vložení zprávy do zprávy výjimky. Vaše aplikace může získat iterátor pro všechny klíče voláním metody `JmsExceptionDetail.getKeys()`.
- Zpráva se vloží do zprávy výjimky. Například vložení zprávy může být název fronty, která způsobila výjimku, a může být užitečné, aby vaše aplikace měla přístup k tomuto názvu. Vaše aplikace může získat vložení zprávy odpovídající zadanému klíči voláním metody `JmsExceptionDetail.getValue()`.

Všechny metody v rozhraní `JmsExceptionDetail` vrátí hodnotu null, pokud nejsou k dispozici žádné podrobnosti.

Pokud se například aplikace pokusí vytvořit producent zpráv pro frontu IBM MQ , která neexistuje, dojde k výjimce 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.DetailedInvalidDestinationException`je podtřídou následující třídy a implementuje rozhraní `com.ibm.msg.client.jms.JmsExceptionDetail` .

- V 9.3.0 JM 3.0 V 9.3.0 `jakarta.jms.InvalidDestinationException`
- JMS 2.0 `javax.jms.InvalidDestinationException`

Propojené výjimky

Propojená výjimka poskytuje další informace o běhovém problému. Proto by měla aplikace pro každou vyvolanou výjimku `JMSException` zkontrolovat propojenou výjimku.

Samotná propojená výjimka může mít jinou propojenou výjimku, a proto propojené výjimky tvoří řetězec, který vede zpět k původnímu základnímu problému. Propojená výjimka je implementována pomocí mechanizmu zřetězených výjimek třídy `java.lang.Throwable` a vaše aplikace může získat propojenou

výjimku voláním metody `Throwable.getCause()` . V případě metody `JMSException`se metoda `getLinkedException()` deleguje na metodu `Throwable.getCause()` .

Pokud například aplikace při připojování ke správci front zadá nesprávné číslo portu, budou výjimky tvořit následující řetězec:

```
com.ibm.msg.client.jms.DetailIllegalStateException
|
+--->
    com.ibm.mq.MQException
    |
    +--->
        com.ibm.mq.jmqi.JmqiException
        |
        +--->
            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 vyvolány následujícími vrstvami:

- První výjimka, instance podtřídy `JMSException`, je vyvolána společnou vrstvou v souboru IBM MQ classes for JMS.
- Další výjimku, instanci `com.ibm.mq.MQException`, vygeneruje poskytovatel systému zpráv IBM MQ .
- Další dvě výjimky, z nichž obě jsou instancemi produktu `com.ibm.mq.jmqi.JmqiException`, jsou vyvolány rozhraním JMQI (Java Message Queueing Interface). Rozhraní JMQI je komponenta, kterou produkt IBM MQ classes for JMS používá ke komunikaci se správcem front.
- Poslední výjimka, instance `java.net.ConnectionException`, je vyvolána knihovnou tříd Java .

Další informace o vrstvené architektuře produktu IBM MQ classes for JMSnaleznete v tématu IBM MQ pro architekturu JMS.

Svou aplikaci můžete kódovat tak, aby procházením tohoto řetězce extrahovala všechny příslušné informace, jak ukazuje následující příklad:

V 9.3.0 JM 3.0 V 9.3.0

```
import com.ibm.msg.client.jms.JmsExceptionDetail;
import com.ibm.mq.MQException;
import com.ibm.mq.jmqi.JmqiException;
import jakarta.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();
  }
}
```

JMS 2.0

```
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();
  }
}
```

Všimněte si, že vaše aplikace by měla vždy zkontrolovat typ každé výjimky v řetězci, protože typ výjimky se může lišit a výjimky různých typů zapouzdřují různé informace.

Získání IBM MQ specifických informací o problému

Instance `com.ibm.mq.MQException` a `com.ibm.mq.jmqi.JmqiException` zapouzdřit IBM MQ specifické informace o problému.

`MQException` zapouzdřuje následující informace:

- Kód dokončení, který může aplikace získat voláním metody `getCompCode()` .
- Kód příčiny, který může aplikace získat voláním metody `getReason()` .

Příklady použití těchto metod naleznete v ukázkovém kódu v části propojené výjimky.

`JmqiException` také zapouzdřuje kód dokončení a kód příčiny. Kromě toho obsahuje `JmqiException` informace ve zprávě AMQ *nnnn* nebo CSQ *nnnn* , pokud je nějaká přidružena k výjimce. Aplikace může získat různé komponenty této zprávy voláním následujících metod:

- Metoda `getWmqMsgExplanation()` vrátí vysvětlení zprávy AMQ *nnnn* nebo CSQ *nnnn* .

- Metoda `getWmqMsgSeverity()` vrátí závažnost zprávy AMQ *nnnn* nebo CSQ *nnnn* .
- Metoda `getWmqMsgSummary()` vrátí souhrn zprávy AMQ *nnnn* nebo CSQ *nnnn* .
- Metoda `getWmqMsgUserResponse()` vrátí odezvu uživatele, která je přidružena ke zprávě AMQ *nnnn* nebo CSQ *nnnn* .

Zpracování nekontrolovaných výjimek
Metody na rozhraních, která jsou definována v rozhraní JMS 2.0 , generují nekontrolované výjimky. Základní třída pro tyto výjimky je `JMSRuntimeException`. Proto zachycení `JMSRuntimeExceptions` poskytuje generický způsob zacházení s těmito typy výjimek.

Každý `JMSRuntimeException` zapouzdřuje následující informace:

- Zpráva výjimky specifická pro poskytovatele, kterou může aplikace získat voláním metody `JMSRuntimeException.getMessage()` .
- Kód chyby specifický pro poskytovatele, který může aplikace získat voláním metody `JMSRuntimeException.getErrorCode()` .
- Připojená výjimka. Výjimka vyvolaná voláním rozhraní API JMS 2.0 je často výsledkem problému nižší úrovně, který je ohlášen jinou výjimkou, která je propojena s touto výjimkou. Vaše aplikace může získat připojenou výjimku voláním metody `JMSRuntimeException.getCause()` .

Když voláte metody na rozhraních, která jsou poskytována rozhraním API JMS 2.0 , většina výjimek, které jsou vyvolány IBM MQ classes for JMS , jsou instance podtříd `JMSRuntimeException`. Tyto podtřídy implementují rozhraní `com.ibm.msg.client.jms.JmsExceptionDetail` , které poskytuje následující další informace:

- Vysvětlení zprávy výjimky. Aplikace může tuto zprávu získat voláním metody `JmsExceptionDetail.getExplanation()` .
- Doporučená odezva uživatele na výjimku. Aplikace může tuto zprávu získat voláním metody `JmsExceptionDetail.getUserAction()` .
- Klíče pro vložení zprávy do zprávy výjimky. Vaše aplikace může získat iterátor pro všechny klíče voláním metody `JmsExceptionDetail.getKeys()` .
- Zpráva se vloží do zprávy výjimky. Například vložení zprávy může být název fronty, která způsobila výjimku, a může být užitečné, aby vaše aplikace měla přístup k tomuto názvu. Vaše aplikace může získat vložení zprávy odpovídající zadanému klíči voláním metody `JmsExceptionDetail.getValue()` .

Všechny metody v rozhraní `JmsExceptionDetail` vrátí hodnotu null, pokud nejsou k dispozici žádné podrobnosti.

Pokud se například aplikace pokusí vytvořit `JMSProducer` pro frontu IBM MQ , která neexistuje, dojde k výjimce 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.DetailedInvalidDestinationException`je podtřídou následující třídy a implementuje rozhraní `com.ibm.msg.client.jms.JmsExceptionDetail` .

- V 9.3.0 JM 3.0 V 9.3.0 `jakarta.jms.InvalidDestinationException`
- JMS 2.0 `javax.jms.InvalidDestinationException`

Zřetězené výjimky

Obvykle jsou výjimky způsobeny jinými výjimkami. Proto by měla vaše aplikace pro každou vyvolanou výjimku `JMSRuntimeException` zkontrolovat připojenou výjimku.

Příčinou `JMSRuntimeException` může být další výjimka. Tyto výjimky tvoří řetězec, který vede zpět k původnímu problému. Příčina výjimky je implementována pomocí mechanizmu zřetězených výjimek třídy `java.lang.Throwable` a vaše aplikace může získat propojenou výjimku voláním metody `Throwable.getCause()` .

Pokud například aplikace při připojování ke správci front zadá nesprávné číslo portu, budou výjimky tvořit následující řetězec:

```
com.ibm.msg.client.jms.DetailIllegalStateException
|
+--->
    com.ibm.mq.MQException
    |
    +--->
        com.ibm.mq.jmqi.JmqiException
        |
        +--->
            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 vyvolány následujícími vrstvami:

- První výjimka, instance podtřídy `JMSRuntimeException`, je vyvolána společnou vrstvou v souboru IBM MQ classes for JMS.
- Další výjimku, instanci `com.ibm.mq.MQException`, vygeneruje poskytovatel systému zpráv IBM MQ .
- Další dvě výjimky, z nichž obě jsou instancemi produktu `com.ibm.mq.jmqi.JmqiException`, jsou vyvolány rozhraním JMQI (Java Message Queueing Interface). Rozhraní JMQI je komponenta, kterou produkt IBM MQ classes for JMS používá ke komunikaci se správcem front.
- Poslední výjimka, instance `java.net.ConnectionException`, je vyvolána knihovnou tříd Java .

Další informace o vrstvené architektuře produktu IBM MQ classes for JMSnaleznete v tématu IBM MQ pro architekturu JMS.

Svou aplikaci můžete kódovat tak, aby procházením tohoto řetězce extrahovala všechny příslušné informace, jak ukazuje následující příklad:

V 9.3.0 JM 3.0 V 9.3.0

```
import com.ibm.msg.client.jms.JmsExceptionDetail;
import com.ibm.mq.MQException;
import com.ibm.mq.jmqi.JmqiException;
import jakarta.jms.JMSRuntimeException;
.
.
.
catch (JMSRuntimeException je) {
  System.err.println("Caught JMSRuntimeException");
  // Check for linked exceptions in JMSRuntimeException
  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 JMSRuntimeException) {
      JMSRuntimeException je1 = (JMSRuntimeException) 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();
  }
}
```

JMS 2.0

```
import com.ibm.msg.client.jms.JmsExceptionDetail;
import com.ibm.mq.MQException;
import com.ibm.mq.jmqi.JmqiException;
import javax.jms.JMSRuntimeException;
.
.
.
catch (JMSRuntimeException je) {
  System.err.println("Caught JMSRuntimeException");
  // Check for linked exceptions in JMSRuntimeException
  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 JMSRuntimeException) {
      JMSRuntimeException je1 = (JMSRuntimeException) 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();
  }
}
```

Všimněte si, že vaše aplikace by měla vždy zkontrolovat typ každé výjimky v řetězci, protože typ výjimky se může lišit a výjimky různých typů zapouzdřují různé informace.

Získání IBM MQ specifických informací o problému

Instance `com.ibm.mq.MQException` a `com.ibm.mq.jmqi.JmqiException` zapouzdřit IBM MQ specifické informace o problému.

`MQException` zapouzdřuje následující informace:

- Kód dokončení, který může aplikace získat voláním metody `getCompCode()` .
- Kód příčiny, který může aplikace získat voláním metody `getReason()` .

Příklady použití těchto metod naleznete v ukázkovém kódu v části Zřetězené výjimky.

`JmqiException` také zapouzdřuje kód dokončení a kód příčiny. Kromě toho obsahuje `JmqiException` informace ve zprávě AMQ *nnnn* nebo CSQ *nnnn* , pokud je nějaká přidružena k výjimce. Aplikace může získat různé komponenty této zprávy voláním následujících metod:

- Metoda `getWmqMsgExplanation()` vrátí vysvětlení zprávy AMQ *nnnn* nebo CSQ *nnnn* .
- Metoda `getWmqMsgSeverity()` vrátí závažnost zprávy AMQ *nnnn* nebo CSQ *nnnn* .
- Metoda `getWmqMsgSummary()` vrátí souhrn zprávy AMQ *nnnn* nebo CSQ *nnnn* .
- Metoda `getWmqMsgUserResponse()` vrátí odezvu uživatele, která je přidružena ke zprávě AMQ *nnnn* nebo CSQ *nnnn* .

ExceptionListeners
K objektům JMS `Connection` a `JMSContext` je přidruženo připojení ke správci front. Vaše aplikace může registrovat `ExceptionListener` s JMS `Connection` nebo `JMSContext`. Dojde-li k problému, který učiní připojení přidružené k `Connection` nebo `JMSContext` nepoužitelným, produkt IBM MQ classes for JMS doručí výjimku `ExceptionListener` voláním jeho metody `onException()` . Vaše aplikace pak bude mít možnost znovu navázat spojení.

Produkt IBM MQ classes for JMS může také doručit výjimku do modulu listener pro výjimky, pokud dojde k problému při asynchronním pokusu o doručení zprávy.

Listenery výjimek

V produktu IBM MQ 8.0.0 Fix Pack 2je výchozí hodnota vlastnosti `ConnectionFactory` ASYNCEXCEPTION změněna na hodnotu ASYNC_EXCEPTIONS_CONNECTIONBROKEN, aby se zachovalo chování pro aktuální aplikace JMS, které konfigurují agenta JMS `MessageListener` a agenta JMS `ExceptionListener`, a aby se zajistilo, že IBM MQ classes for JMS jsou konzistentní se specifikací JMS . V důsledku toho jsou produktu `ExceptionListener` aplikace doručeny pouze výjimky, které odpovídají chybovým kódům přerušeného připojení.

APAR IT14820, obsaženo v souboru IBM MQ 9.0.0 Fix Pack 1, aktualizuje IBM MQ classes for JMS tak, aby:

- `ExceptionListener` , který je registrován aplikací, je vyvolán pro všechny výjimky s přerušením připojení, bez ohledu na to, zda aplikace používá synchronní nebo asynchronní spotřebitele zpráv.
- Výjimky bez přerušení připojení (například MQRC_GET_INHIBITED), které se vyskytnou během doručování zpráv, jsou doručeny do `ExceptionListener` aplikace, když aplikace používá asynchronní spotřebitele zpráv, a JMS ConnectionFactory , kterou používá aplikace, má vlastnost ASYNC_EXCEPTION nastavenou na hodnotu ASYNC_EXCEPTIONS_ALL.

Poznámka: `ExceptionListener` je vyvolán pouze jednou pro výjimku přerušení připojení, a to i v případě, že jsou přerušena dvě připojení TCP/IP (jedno používané připojením JMS a jedno používané relací JMS).

V případě jakéhokoli jiného typu problému je vyvolána výjimka aktuálním voláním rozhraní API JMS . Typ vyvolané výjimky závisí na verzi rozhraní API JMS , které aplikace používá:

- Pokud aplikace používá rozhraní, která jsou poskytována specifikací JMS 1.1 , výjimkou je `JMSException`. Další informace o způsobu zpracování těchto výjimek naleznete v tématu “Zpracování kontrolovaných výjimek” na stránce 229.
- Pokud aplikace používá rozhraní JMS 2.0 , výjimkou je `JMSRuntimeException`. Další informace o způsobu zpracování těchto výjimek naleznete v tématu “Zpracování nekontrolovaných výjimek” na stránce 232.

Pokud aplikace neregistrují modul listener pro výjimky s `Connection` nebo `JMSContext`, všechny výjimky, které by byly doručeny modulu listener pro výjimky, se zapíší do protokolu IBM MQ classes for JMS .

Přístup k funkcím IBM MQ z aplikace IBM MQ classes for JMS

Produkt IBM MQ classes for JMS poskytuje prostředky pro využití řady funkcí produktu IBM MQ.

Upozornění: Tyto funkce jsou mimo specifikaci JMS nebo v určitých případech porušují specifikaci JMS . Pokud je použijete, je nepravděpodobné, že by vaše aplikace byla kompatibilní s ostatními poskytovateli JMS . Funkce, které nejsou v souladu se specifikací JMS , jsou označeny upozorněním na upozornění.

Čtení a zápis deskriptoru zprávy z aplikace IBM MQ classes for JMS
Schopnost přístupu k deskriptoru zpráv (MQMD) můžete řídit nastavením vlastností pro cíl a zprávu.

Některé aplikace systému IBM MQ vyžadují nastavení specifických hodnot v deskriptoru MQMD pro zprávy, které jim byly odeslány. Produkt IBM MQ classes for JMS poskytuje atributy zpráv, které umožňují aplikacím JMS nastavit pole MQMD, a povolit tak aplikacím JMS "řídit" aplikace IBM MQ .

Musíte nastavit vlastnost cílového objektu WMQ_MQ_MD_WRITE_ENABLED na hodnotu true, aby se nastavení vlastností MQMD projevila. Poté můžete k přiřazení hodnot k polím MQMD použít metody nastavení vlastností zprávy (například vlastnost setString). Všechna pole MQMD jsou vystavena s výjimkou polí StrucId a Version; pole BackoutCount lze číst, ale nelze do něj zapisovat.

Tento příklad má za následek vložení zprávy do fronty nebo tématu s produktem 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 JMS_IBM_MQMD_UserIdentifierje nutné nastavit WMQ_MQMD_MESSAGE_CONTEXT. Další informace o použití příkazu WMQ_MQMD_MESSAGE_CONTEXT viz “Vlastnosti objektu zprávy JMS” na stránce 238.

Podobně můžete extrahovat obsah polí MQMD nastavením parametru WMQ_MQMD_READ_ENABLED na hodnotu true před přijetím zprávy a následným použitím metod get zprávy, například vlastnosti getString. Všechny přijaté vlastnosti jsou jen pro čtení.

Tento příklad způsobí, že pole *value* bude obsahovat hodnotu MQMD.ApplIdentityData zprávy získané 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 cílového objektu JMS
Dvě vlastnosti cílového objektu řídí přístup k MQMD z produktu JMSa třetí řídí kontext zprávy.

Tabulka 36. Názvy a popisy vlastností

Vlastnost	Zkrácená forma	Popis
WMQ_MQMD_WRITE_ENABLED	MDW	Zda může aplikace JMS nastavit hodnoty polí MQMD
Parametr 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 se tato vlastnost projevila, musí být aplikace spuštěna s příslušným kontextovým oprávněním.

Tabulka 37. Názvy vlastností, hodnoty a metody set

Vlastnost	Platné hodnoty v nástroji pro administraci (předvolba je tučně)	Platné hodnoty v programech	Nastavit metodu
WMQ_MQMD_WRITE POVOLENO	• **No** Všechny vlastnosti JMS_IBM_MQMD* jsou ignorovány a jejich hodnoty nejsou zkopírovány do základní struktury MQMD. • YES Vlastnosti JMS_IBM_MQMD* jsou zpracovány. Jejich hodnoty jsou zkopírovány do podkladové struktury MQMD.	• **Nepravda** • Pravda	setMQMDWritepovoleno

Tabulka 37. Názvy vlastností, hodnoty a metody set (pokračování)			
Vlastnost	**Platné hodnoty v nástroji pro administraci (předvolba je tučně)**	**Platné hodnoty v programech**	**Nastavit metodu**
WMQ_MQMD_READ_ENABLED	• **No** Při odesílání zpráv nejsou vlastnosti JMS_IBM_MQMD* pro odeslanou zprávu aktualizovány tak, aby odrážely aktualizované hodnoty polí v deskriptoru 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* pro odeslanou zprávu aktualizovány tak, aby odrážely aktualizované hodnoty polí v deskriptoru MQMD, včetně těch, které odesilatel explicitně nenastavil. 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 API MQOPEN a struktura MQPMO neurčují žádné explicitní volby kontextu zprávy. • KONTEX_IDENTITY_SET_IDENTITY_CONTEXT Volání rozhraní API MQOPEN určuje volbu kontextu zprávy MQOO_SET_IDENTITY_CONTEXT a struktura MQPMO určuje volbu MQPMO_SET_IDENTITY_CONTEXT • NASTAVIT_VŠECHNY_KONTEXT Volání MQOPEN API určuje volbu kontextu zprávy MQOO_SET_ALL_CONTEXT a struktura MQPMO určuje volbu MQPMO_SET_ALL_CONTEXT	• **WMQ_MDCTX_DEFAULT** • WMQ_MDCTX_SET_IDENTITY_CONTEXT • WMQ_MDCTX_SET_ALL_CONTEXT	Kontext setMQMDMessage

Vlastnosti objektu zprávy JMS
Vlastnosti objektu zprávy s předponou JMS_IBM_MQMD vám umožňují nastavit nebo číst odpovídající pole MQMD.

Odesílání zpráv

Jsou reprezentována všechna pole MQMD s výjimkou polí StrucId a Version. Tyto vlastnosti odkazují pouze na pole MQMD. V případě, že se vlastnost vyskytuje jak v záhlaví MQMD, tak v záhlaví MQRFH2 , není verze v MQRFH2 nastavena nebo extrahována.

Lze nastavit libovolnou z těchto vlastností s výjimkou JMS_IBM_MQMD_BackoutCount. Jakákoli hodnota nastavená pro JMS_IBM_MQMD_BackoutCount je ignorována.

Pokud má vlastnost maximální délku a zadáte hodnotu, která je příliš dlouhá, hodnota se ořízne.

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 parametr WMQ_MQMD_MESSAGE_CONTEXT na odpovídající hodnotu, bude hodnota vlastnosti ignorována. Nastavíte-li parametr WMQ_MQMD_MESSAGE_CONTEXT na odpovídající hodnotu, ale nemáte dostatečná kontextová oprávnění pro správce front, bude vydána výjimka JMSException. Vlastnosti vyžadující specifické hodnoty WMQ_MQMD_MESSAGE_CONTEXT jsou následující.

Následující vlastnosti vyžadují nastavení parametru WMQ_MQMD_MESSAGE_CONTEXT na hodnotu WMQ_MDCTX_SET_IDENTITY_CONTEXT nebo WMQ_MDCTX_SET_ALL_CONTEXT:

- JMS_IBM_MQMD_UserIdentifier
- JMS_IBM_MQMD_AccountingToken
- Data JMS_IBM_MQMD_ApplIdentity

Následující vlastnosti vyžadují nastavení parametru WMQ_MQMD_MESSAGE_CONTEXT na hodnotu WMQ_MDCTX_SET_ALL_CONTEXT:

- Typ JMS_IBM_MQMD_PutAppl
- Název JMS_IBM_MQMD_PutAppl
- JMS_IBM_MQMD_PutDate
- JMS_IBM_MQMD_PutTime
- Data JMS_IBM_MQMD_ApplOrigin

Příjem zpráv

Všechny tyto vlastnosti jsou k dispozici pro přijatou zprávu, pokud je vlastnost WMQ_MQMD_READ_ENABLED nastavena na hodnotu true, bez ohledu na skutečné vlastnosti, které nastavila produkující aplikace. Aplikace nemůže upravit vlastnosti přijaté zprávy, pokud nejsou nejprve vymazány všechny vlastnosti v souladu se specifikací JMS . Přijatá zpráva může být postoupena bez úpravy vlastností.

Upozornění: Pokud aplikace obdrží zprávu z místa určení s vlastností WMQ_MQMD_READ_ENABLED nastavenou na hodnotu true a předá ji do místa určení s hodnotou WMQ_MQMD_WRITE_ENABLED nastavenou na hodnotu true, bude výsledkem zkopírování všech hodnot polí MQMD přijaté zprávy do předané zprávy.

Tabulka vlastností

V této tabulce jsou uvedeny vlastnosti objektu zprávy reprezentující pole MQMD. Úplný popis polí a jejich přípustných hodnot naleznete v odkazech.

Tabulka 38. Názvy, popisy a typy vlastností

Vlastnost	Popis	Java Typ	Odkaz na úplný popis
JMS_IBM_MQMD_Report	Volby pro zprávy sestavy	Celé číslo	Sestava
JMS_IBM_MQMD_MsgType	Typ zprávy	Celé číslo	MsgType
Vypršení platnosti JMS_IBM_MQMD_Expiry	Životnost zprávy	Celé číslo	Vypršení

Tabulka 38. Názvy, popisy a typy vlastností (pokračování)

Vlastnost	Popis	Java Typ	Odkaz na úplný popis
Zpětná vazba JMS_IBM_MQMD_Feedback	Zpětná vazba nebo kód příčiny	Celé číslo	Zpětná vazba
Kódování JMS_IBM_MQMD_Encoding	Číselné kódování dat zprávy	Celé číslo	Kódování
JMS_IBM_MQMD_CodedCharSetId	Identifikátor znakové sady dat zprávy	Celé číslo	CodedCharSetId
Formát JMS_IBM_MQMD_Format	Název formátu dat zprávy	Řetězec	Formát
Priorita JMS_IBM_MQMD_Priority [1]	Priorita zprávy	Celé číslo	Priorita
Persistence JMS_IBM_MQMD_Persistence	Trvalost zpráv	Celé číslo	Trvání
JMS_IBM_MQMD_MsgId [2]	Identifikátor zprávy	Objekt (bajt []) [4]	MsgId
JMS_IBM_MQMD_CorrelId [3]	Identifikátor korelace	Objekt (bajt []) [4]	CorrelId
JMS_IBM_MQMD_BackoutCount	Počítadlo odvolaných	Celé číslo	BackoutCount
JMS_IBM_MQMD_ReplyToQ	Název fronty odpovědí	Řetězec	ReplyToQ
JMS_IBM_MQMD_ReplyToQMgr	Název správce front odpovědí	Řetězec	ReplyToQMgr
JMS_IBM_MQMD_UserIdentifier	Identifikátor uživatele	Řetězec	UserIdentifier
JMS_IBM_MQMD_AccountingToken	Token evidence	Objekt (bajt []) [4]	AccountingToken
Data JMS_IBM_MQMD_ApplIdentity	Údaje aplikace týkající se identity	Řetězec	ApplIdentityData
Typ JMS_IBM_MQMD_PutAppl	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
Data JMS_IBM_MQMD_ApplOrigin	Údaje o žádosti týkající se původu	Řetězec	ApplOriginData
JMS_IBM_MQMD_GroupId	Identifikátor skupiny	Objekt (bajt []) [4]	GroupId
JMS_IBM_MQMD_MsgSeqPočet	Pořadové číslo logické zprávy ve skupině	Celé číslo	MsgSeqNumber
Posunutí JMS_IBM_MQMD_Offset	Posunutí dat ve fyzické zprávě od začátku logické zprávy	Celé číslo	Offset

Tabulka 38. Názvy, popisy a typy vlastností (pokračování)			
Vlastnost	**Popis**	**Java Typ**	**Odkaz na úplný popis**
JMS_IBM_MQMD_MsgFlags	Příznaky zprávy	Celé číslo	MsgFlags
JMS_IBM_MQMD_OriginalLength	Délka původní zprávy	Celé číslo	OriginalLength

1. **Upozornění:** Pokud přiřadíte hodnotu JMS_IBM_MQMD_Priority, která není v rozsahu 0-9, porušuje to 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 JMS_IBM_MQMD_MsgId, tato hodnota se zkopíruje do JMSMessageID. Proto není nastaven poskytovatelem JMS a nemusí být jedinečný: to porušuje specifikaci JMS .
3. **Upozornění:** Pokud přiřadíte hodnotu JMS_IBM_MQMD_CorrelId , která začíná řetězcem 'ID:', porušuje to specifikaci JMS .
4. **Upozornění:** Použití vlastností bajtového pole ve zprávě porušuje specifikaci JMS .

Přístup k IBM MQ datům zprávy z aplikace pomocí IBM MQ classes for JMS
K úplným datům zprávy produktu IBM MQ v rámci aplikace můžete přistupovat pomocí produktu IBM MQ classes for JMS. Chcete-li získat přístup ke všem datům, zpráva musí být JMSBytesMessage. Tělo JMSBytesMessage zahrnuje libovolné záhlaví MQRFH2 , jakákoli jiná záhlaví IBM MQ a následující data zprávy.

Nastavte vlastnost WMQ_MESSAGE_BODY místa určení na hodnotu WMQ_MESSAGE_BODY_MQ, chcete-li přijmout všechna data těla zprávy v souboru JMSBytesMessage.

Je-li volba WMQ_MESSAGE_BODY nastavena na hodnotu WMQ_MESSAGE_BODY_JMS nebo WMQ_MESSAGE_BODY_UNSPECIFIED, bude tělo zprávy vráceno bez záhlaví JMS MQRFH2 a vlastnosti JMSBytesMessage budou odrážet vlastnosti nastavené v souboru RFH2.

Některé aplikace nemohou používat funkce popsané v tomto tématu. Pokud je aplikace připojena ke správci front IBM MQ V6 nebo pokud má nastavenu volbu PROVIDERVERSION na hodnotu 6, funkce nejsou k dispozici.

Odeslání zprávy

Při odesílání zpráv má vlastnost místa určení WMQ_MESSAGE_BODYpřednost před vlastností WMQ_TARGET_CLIENT.

Je-li volba WMQ_MESSAGE_BODY nastavena na hodnotu WMQ_MESSAGE_BODY_JMS, produkt IBM MQ classes for JMS automaticky vygeneruje záhlaví MQRFH2 na základě nastavení vlastností a polí záhlaví JMSMessage .

Je-li volba WMQ_MESSAGE_BODY nastavena na hodnotu WMQ_MESSAGE_BODY_MQ, do těla zprávy se nepřidá žádné další záhlaví.

Je-li parametr WMQ_MESSAGE_BODY nastaven na hodnotu WMQ_MESSAGE_BODY_UNSPECIFIED, IBM MQ classes for JMS odešle záhlaví MQRFH2 , pokud není parametr WMQ_TARGET_CLIENT nastaven na hodnotu WMQ_TARGET_DEST_MQ. Při příjmu má nastavení parametru WMQ_TARGET_CLIENT na hodnotu WMQ_TARGET_DEST_MQ za následek odebrání MQRFH2 z těla zprávy.

Poznámka: JMSBytesMessage a JMSTextMessage nevyžadují MQRFH2, zatímco JMSStreamMessage, JMSMapMessagea JMSObjectMessage ano.

WMQ_MESSAGE_BODY_UNSPECIFIED je výchozí nastavení pro WMQ_MESSAGE_BODYa WMQ_TARGET_DEST_JMS je výchozí nastavení pro WMQ_TARGET_CLIENT.

Pokud odešlete zprávu JMSBytesMessage, můžete přepsat výchozí nastavení pro tělo zprávy JMS při vytváření zprávy 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 JMS , pokud neexistuje předchozí záhlaví WebSphere MQ .
- JMS_IBM_Character_Set nebo JMS_IBM_MQMD_CodedCharSetId: Tato vlastnost určuje CCSID záhlaví IBM MQ nebo informačního obsahu aplikace, která spouští tělo zprávy JMS , pokud neexistuje předchozí záhlaví WebSphere MQ .
- JMS_IBM_Encoding nebo JMS_IBM_MQMD_Encoding: Tato vlastnost určuje kódování informačního obsahu záhlaví nebo aplikace IBM MQ , které spustí tělo zprávy JMS , pokud před záhlavím produktu WebSphere MQ neexistuje.

Pokud jsou zadány oba typy vlastností, vlastnosti JMS_IBM_MQMD_* potlačí příslušné vlastnosti JMS_IBM_* , pokud je vlastnost místa určení WMQ_MQMD_WRITE_ENABLED nastavena na hodnotu true.

Rozdíly v platnosti mezi nastavením vlastností zprávy pomocí JMS_IBM_MQMD_* a JMS_IBM_* jsou významné:

1. Vlastnosti JMS_IBM_MQMD_* jsou specifické pro poskytovatele produktu IBM MQ JMS .
2. Vlastnosti JMS_IBM_MQMD_* jsou nastaveny pouze v souboru MQMD. Vlastnosti JMS_IBM_* jsou nastaveny v souboru 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 souboru JMSMessage.

   Přijímající aplikace pravděpodobně předpokládá, že hodnoty MQMD.Encoding a MQMD.CodedCharSetId odpovídají kódování a znakové sadě čísel a textu v těle zprávy. Jsou-li použity vlastnosti JMS_IBM_MQMD_* , je odpovědností odesílající aplikace, aby tak učinily. Kódování a znaková sada čísel a textu v těle zprávy jsou nastaveny pomocí vlastností JMS_IBM_* .

   Chybně kódovaný úsek kódu v produktu Obrázek 39 na stránce 242 odešle zprávu kódovanou ve znakové sadě 1208 s MQMD.CodedCharSetId nastavenou na 37.

   a. Odeslat chybně zakódovanou 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 s odvoláním na hodnotu JMS_IBM_CHARACTER_SET nastavenou hodnotou MQMD.CodedCharSetId:

   ```
   TextMessage tmi = (TextMessage) cons.receive();
   System.out.println("Message is \"" + tmi.getText() + "\"");
   ```

   c. Výsledný výstup:

   ```
   Message is "éÈÊ'>...??>?"
   ```

   Obrázek 39. Nekonzistentně kódované MQMD a data zpráv

Kterýkoli z úseků kódu v produktu Obrázek 40 na stránce 243 má za následek vložení zprávy do fronty nebo tématu s tělem, které obsahuje informační obsah aplikace, aniž by bylo přidáno automaticky generované záhlaví 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 40. Odešlete zprávu s tělem zprávy MQ .

Příjem zprávy

Je-li volba WMQ_MESSAGE_BODY nastavena na hodnotu WMQ_MESSAGE_BODY_JMS, typ a tělo příchozí zprávy JMS jsou určeny obsahem přijaté zprávy WebSphere MQ . Typ zprávy a tělo zprávy jsou určeny poli v záhlaví MQRFH2 nebo v souboru MQMD, pokud neexistuje hodnota 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 MQGET API. Délka těla zprávy je délka vrácená voláním MQGET . Znaková sada a kódování dat v těle zprávy je určeno poli CodedCharSetId a Kódování MQMD. Formát dat v těle zprávy je určen polem Formát konzoly MQMD .

Je-li parametr WMQ_MESSAGE_BODY nastaven na hodnotu WMQ_MESSAGE_BODY_UNSPECIFIED, výchozí hodnota IBM MQ classes for JMS jej nastaví na hodnotu WMQ_MESSAGE_BODY_JMS.

Když obdržíte JMSBytesMessage, můžete jej dekódovat odkazem 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 JMS , pokud neexistuje předchozí záhlaví WebSphere MQ .
- JMS_IBM_Character_Set nebo JMS_IBM_MQMD_CodedCharSetId: Tato vlastnost určuje CCSID záhlaví IBM MQ nebo informačního obsahu aplikace, která spouští tělo zprávy JMS , pokud neexistuje předchozí záhlaví WebSphere MQ .
- JMS_IBM_Encoding nebo JMS_IBM_MQMD_Encoding: Tato vlastnost určuje kódování informačního obsahu záhlaví nebo aplikace IBM MQ , které spustí tělo zprávy JMS , pokud před záhlavím produktu WebSphere MQ neexistuje.

Následující úsek kódu vede k přijaté zprávě, která je JMSBytesMessage. Bez ohledu na obsah přijaté zprávy a pole formátu přijaté zprávy MQMDje zpráva 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 (tj. jako součást těla zprávy JMS).

Tabulka 39. Názvy a popisy vlastností		
Vlastnost	**Zkrácená forma**	**Popis**
TĚLO zprávy WMQ_MESSAGE_BODY	MBODY	Zda aplikace JMS zpracovává MQRFH2 zprávy IBM MQ jako součást informačního obsahu zprávy (tj. jako součást těla zprávy JMS).

Tabulka 40. Názvy vlastností, hodnoty a metody set			
Vlastnost	**Platné hodnoty v nástroji pro administraci (předvolba je tučně)**	**Platné hodnoty v programech**	**Nastavit metodu**
WMQ_MESSAGE_BODY	• **Neuvedeno** Při odesílání produkt IBM MQ classes for JMS generuje nebo negeneruje a nezahrne záhlaví MQRFH2 v závislosti na hodnotě WMQ_TARGET_CLIENT. Při příjmu se chová jako hodnota JMS. • JMS Při odesílání produkt IBM MQ classes for JMS automaticky vygeneruje záhlaví MQRFH2 a zahrne je do zprávy IBM MQ . Při příjmu IBM MQ classes for JMS nastavte vlastnosti zprávy JMS podle hodnot v MQRFH2 (je-li k dispozici); neuvádí MQRFH2 jako součást těla zprávy JMS . • MQ Při odesílání produkt IBM MQ classes for JMS negeneruje položku MQRFH2. Při příjmu produkt IBM MQ classes for JMS prezentuje MQRFH2 jako součást těla zprávy JMS .	• **ZPRÁVA WMQ_MESSAGE_BODY_UNURČENÝ** • WMQ_MESSAGE_BODY_JMS • WMQ_MESSAGE_BODY_MQ	setMessageBodyStyle

JMS trvalých zpráv
Aplikace IBM MQ classes for JMS mohou použít atribut fronty **NonPersistentMessageClass** k zajištění lepšího výkonu pro trvalé zprávy JMS na úkor určité spolehlivosti.

Fronta IBM MQ má atribut s názvem **NonPersistentMessageClass**. Hodnota tohoto atributu určuje, zda jsou přechodné zprávy ve frontě při restartování správce front vyřazeny.

Atribut pro lokální frontu můžete nastavit pomocí příkazu IBM MQ Script (MQSC) DEFINE QLOCAL s jedním z následujících parametrů:

NPMCLASS (NORMÁLNÍ)
Přechodné zprávy ve frontě jsou při restartování správce front zrušeny. Toto je výchozí hodnota.

NPMCLASS (HIGH)
Přechodné zprávy ve frontě nejsou zahozeny, když se správce front restartuje po uvedení do klidového stavu nebo okamžitém vypnutí. Přechodné zprávy mohou být vyřazeny, avšak po preventivním ukončení práce systému nebo selhání.

Toto téma popisuje, jak mohou aplikace IBM MQ classes for JMS používat tento atribut fronty, aby poskytovaly lepší výkon pro JMS trvalé zprávy.

Vlastnost PERSISTENCE objektu Queue nebo Topic může mít hodnotu HIGH. Tuto hodnotu můžete nastavit pomocí administračního nástroje produktu IBM MQ JMS , nebo může aplikace volat metodu Destination.setPersistence() s předáním hodnoty WMQConstants.WMQ_PER_NPHIGH jako parametru.

Pokud aplikace odešle trvalou zprávu JMS nebo dočasnou zprávu JMS do místa určení, kde má vlastnost PERSISTENCE hodnotu HIGH a základní fronta IBM MQ je nastavena na hodnotu NPMCLASS (HIGH), bude zpráva vložena do fronty jako dočasná zpráva IBM MQ . Pokud vlastnost PERSISTENCE místa určení nemá hodnotu HIGH nebo pokud je základní fronta nastavena na hodnotu NPMCLASS (NORMAL), je do fronty vložena trvalá zpráva JMS jako trvalá zpráva IBM MQ a do fronty je vložena dočasná zpráva JMS jako dočasná zpráva IBM MQ .

Pokud je trvalá zpráva systému JMS vložena do fronty jako dočasná zpráva systému IBM MQ a chcete zajistit, aby zpráva nebyla zahozena po uvedení správce front do klidového stavu nebo po jeho okamžitém ukončení, musí být všechny fronty, jejichž prostřednictvím může být zpráva směrována, nastaveny na hodnotu NPMCLASS (HIGH). V doméně publikování/odběru tyto fronty zahrnují fronty odběratele. Jako pomůcku pro vynucení této konfigurace produkt IBM MQ classes for JMS vygeneruje výjimku InvalidDestination, pokud se aplikace pokusí vytvořit spotřebitele zpráv pro místo určení, kde má vlastnost PERSISTENCE hodnotu HIGH a základní fronta IBM MQ je nastavena na hodnotu NPMCLASS (NORMAL).

Nastavení vlastnosti PERSISTENCE místa určení na hodnotu HIGH neovlivní způsob příjmu zprávy z daného místa určení. Zpráva odeslaná jako trvalá zpráva systému JMS je přijata jako trvalá zpráva systému JMS a zpráva odeslaná jako dočasná zpráva systému JMS je přijata jako dočasná zpráva systému JMS .

Když aplikace odešle první zprávu do místa určení, kde má vlastnost PERSISTENCE hodnotu HIGH, nebo když aplikace vytvoří prvního spotřebitele zpráv pro místo určení, kde má vlastnost PERSISTENCE hodnotu HIGH, produkt IBM MQ classes for JMS vydá volání MQINQ s cílem určit, zda je v základní frontě IBM MQ nastavena hodnota NPMCLASS (HIGH). Aplikace proto musí mít oprávnění dotazovat se na frontu. Kromě toho produkt IBM MQ classes for JMS zachová výsledek volání MQINQ až do odstranění místa určení a nevydá další volání MQINQ. Pokud tedy změníte nastavení NPMCLASS v základní frontě, zatímco aplikace stále používá místo určení, produkt IBM MQ classes for JMS si nové nastavení nevšimne.

Povolením vkládání trvalých zpráv JMS do front IBM MQ jako IBM MQ přechodných zpráv získáte výkon na úkor určité spolehlivosti. Požadujete-li maximální spolehlivost pro trvalé zprávy systému JMS , neodesílejte zprávy do místa určení, kde má vlastnost PERSISTENCE hodnotu HIGH.

Vrstva JMS může používat SYSTEM.JMS.TEMPQ.MODEL, namísto SYSTEM.DEFAULT.MODEL.QUEUE. SYSTEM.JMS.TEMPQ.MODEL vytváří trvalé dynamické fronty, které přijímají trvalé zprávy, protože SYSTEM.DEFAULT.MODEL.QUEUE nemůže přijmout trvalé zprávy. Chcete-li použít dočasné fronty k přijetí trvalých zpráv, musíte proto použít SYSTEM.JMS.TEMPQ.MODELnebo změňte modelovou frontu na alternativní frontu dle vašeho výběru.

Použití TLS s IBM MQ classes for JMS
Aplikace 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 volbu TRANSPORT (CLIENT) podporují šifrování TLS. Protokol TLS poskytuje šifrování komunikace, ověřování a integritu zpráv. Obvykle se používá k zabezpečení komunikace mezi libovolnými dvěma rovnocennými partnery na Internetu nebo v rámci intranetu.

Produkt IBM MQ classes for JMS používá produkt Java Secure Socket Extension (JSSE) ke zpracování šifrování TLS, a proto vyžaduje poskytovatele JSSE. Prostředí JVM prostředí JSE v1.4 mají vestavěného poskytovatele prostředí JSSE. Podrobnosti o tom, jak spravovat a ukládat certifikáty, se mohou u jednotlivých poskytovatelů lišit. Informace o tomto tématu naleznete v dokumentaci poskytovatele JSSE.

Tento oddíl předpokládá, že váš poskytovatel JSSE je správně nainstalován a nakonfigurován a že byly nainstalovány vhodné certifikáty a zpřístupněny vašemu poskytovateli JSSE. Nyní můžete pomocí JMSAdmin nastavit řadu administrativních vlastností.

Pokud vaše aplikace IBM MQ classes for JMS používá tabulku CCDT (Client Channel Definition Table) pro připojení ke správci front, postupujte podle části “Použití tabulky definic kanálů klienta s volbou IBM MQ classes for JMS” na stránce 275.

Vlastnost objektu SSLCIPHERSUITE
Nastavením parametru SSLCIPHERSUITE povolíte šifrování TLS u objektu ConnectionFactory .

Chcete-li povolit šifrování TLS v objektu ConnectionFactory , nastavte pomocí správce JMSAdmin vlastnost SSLCIPHERSUITE na hodnotu CipherSuite podporovanou poskytovatelem JSSE. Musí se shodovat se specifikací CipherSpec nastavenou na cílovém kanálu. CipherSuites se však liší od specifikací CipherSpecs , a mají proto různé názvy. “TLS CipherSpecs a CipherSuites v adresáři IBM MQ classes for JMS” na stránce 249 obsahuje tabulku mapující CipherSpecs podporované produktem IBM MQ na ekvivalentní CipherSuites známé prostředí JSSE. Další informace o specifikacích CipherSpecs a CipherSuites s produktem IBM MQviz Zabezpečení IBM MQ.

Chcete-li například nastavit objekt ConnectionFactory , který lze použít k vytvoření připojení prostřednictvím kanálu MQI s povoleným zabezpečením TLS s CipherSpec TLS_RSA_WITH_AES_128_CBC_SHA, zadejte pro správce JMSAdmin následující příkaz:

```
ALTER CF(my.cf) SSLCIPHERSUITE(SSL_RSA_WITH_AES_128_CBC_SHA)
```

Lze ji také nastavit z aplikace pomocí metody setSSLCipherSuite () na objektu MQConnectionFactory .

Pokud je ve vlastnosti SSLCIPHERSUITE zadána volba CipherSpec , pokusí se správce JMSAdmin mapovat položku CipherSpec na příslušnou sadu CipherSuite a vydá varování. Tento pokus o mapování se neprovede, pokud je vlastnost určena aplikací.

Případně použijte tabulku CCDT (Client Channel Definition Table). Další informace viz téma “Použití tabulky definic kanálů klienta s volbou IBM MQ classes for JMS” na stránce 275.

SSLFIPSREQUIRED vlastnost objektu
Pokud požadujete připojení k použití CipherSuite , která je podporována poskytovatelem JSSE FIPS produktu IBM Java (IBMJSSEFIPS), nastavte vlastnost SSLFIPSREQUIRED továrny připojení na hodnotu YES.

Poznámka: V systému AIX, Linux, and Windowsposkytuje produkt IBM MQ kompatibilitu se standardem FIPS 140-2 prostřednictvím šifrovacího modulu IBM Crypto for C (ICC) . Certifikát pro tento modul byl přesunut do historického stavu. Zákazníci by si měli prohlédnout IBM Crypto for C (ICC) certifikát a měli by si být vědomi všech doporučení poskytnutých NIST. Náhradní modul FIPS 140-3 momentálně probíhá a jeho stav lze zobrazit jeho vyhledáním v modulech NIST CMVP v seznamu procesů.

Výchozí hodnota této vlastnosti je NO, což znamená, že připojení může používat libovolnou sadu CipherSuite podporovanou produktem 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, která se používá k vytvoření následného připojení, je ignorována. Chcete-li použít jinou hodnotu SSLFIPSREQUIRED, musíte restartovat aplikaci.

Aplikace může tuto vlastnost nastavit voláním metody setSSLFipsRequired () objektu ConnectionFactory . Vlastnost je ignorována, pokud není nastavena žádná sada CipherSuite .

Související úlohy

Určení, že za běhu jsou v klientu MQI použity pouze specifikace CipherSpecs s certifikací FIPS.

Související odkazy

Standard FIPS (Federal Information Processing Standards) pro AIX, 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 JMS může zajistit připojení ke správnému správci front zadáním vzoru rozlišujícího názvu (DN). Připojení je úspěšné pouze v případě, že správce front předloží DN, které odpovídá vzoru. Další podrobnosti o formátu tohoto vzoru viz související témata.

DN je nastaveno pomocí vlastnosti SSLPEERNAME objektu ConnectionFactory . Například následující příkaz JMSAdmin nastaví objekt ConnectionFactory tak, aby očekával, že se správce front identifikuje s obecným názvem začínajícím znaky `QMGR.` a s alespoň dvěma názvy organizačních jednotek, z nichž první musí být IBM a druhý WEBSPHERE:

```
ALTER CF(my.cf) SSLPEERNAME(CN=QMGR.*, OU=IBM, OU=WEBSPHERE)
```

Kontrola nerozlišuje velká a malá písmena a místo čárek lze použít středníky. SSLPEERNAME lze také nastavit z aplikace pomocí metody setSSLPeerName () na objektu MQConnectionFactory . Není-li tato vlastnost nastavena, neprovádí se žádná kontrola rozlišujícího názvu dodaného správcem front. Tato vlastnost je ignorována, pokud není nastavena žádná sada CipherSuite .

Vlastnost objektu SSLCERTSTORES

Použijte SSLCERTSTORES k určení seznamu serverů LDAP, které se mají použít pro kontrolu seznamu odvolaných certifikátů (CRL).

Je běžné používat seznam odvolaných certifikátů (CRL) k identifikaci certifikátů, které již nejsou důvěryhodné. Seznamy CRL jsou obvykle hostovány na serverech LDAP. Produkt JMS umožňuje zadat server LDAP pro kontrolu CRL pod Java 2 v1.4 nebo novější. Následující příklad JMSAdmin nařizuje produktu JMS používat seznam CRL hostovaný na serveru LDAP s názvem crl1.ibm.com:

```
ALTER CF(my.cf) SSLCRL(ldap://crl1.ibm.com)
```

Poznámka: Chcete-li úspěšně použít CertStore se seznamem CRL hostovaným na serveru LDAP, ujistěte se, že je vaše sada Java Software Development Kit (SDK) kompatibilní se seznamem CRL. Některé sady SDK vyžadují, aby seznam CRL odpovídal standardu RFC 2587, který definuje schéma pro protokol LDAP v2. Většina serverů LDAP v3 místo toho používá RFC 2256.

Pokud váš server LDAP neběží na výchozím portu 389, můžete zadat port tak, že k názvu hostitele připojíte dvojtečku (:) a číslo portu. Pokud je certifikát předložený správcem front přítomen v seznamu CRL, jehož hostitelem je crl1.ibm.com, není připojení dokončeno. Chcete-li se vyhnout jedinému bodu selhání, produkt JMS umožňuje dodání více serverů LDAP zadáním seznamu 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 postupně pokouší o každý z nich, dokud nenajde server, se kterým by mohl úspěšně ověřit certifikát správce front. Každý server musí obsahovat identické informace.

Řetězec v tomto formátu může být dodán aplikací v metodě MQConnectionFactory.setSSLCertStores (). Alternativně může aplikace vytvořit jeden nebo více objektů java.security.cert.CertStore , umístit je do vhodného objektu Collection a dodat tento objekt Collection do metody setSSLCertStores (). Tímto způsobem může aplikace upravit kontrolu CRL. Podrobné informace o konstrukci a použití objektů CertStore naleznete v dokumentaci JSSE.

Certifikát předložený správcem front při nastavování připojení je ověřen následujícím způsobem:

1. První objekt CertStore v kolekci určené úložištěm sslCertse používá k identifikaci serveru CRL.
2. Došlo k pokusu o kontaktování serveru CRL.
3. Pokud je pokus úspěšný, server vyhledá shodu pro certifikát.
   a. Pokud se zjistí, že je certifikát odvolán, proces vyhledávání je ukončen a požadavek na připojení se nezdaří s kódem příčiny MQRC_SSL_CERTIFICATE_ODVOLÁN.
   b. Není-li certifikát nalezen, proces vyhledávání je ukončen a připojení může pokračovat.
4. Pokud je pokus o kontaktování serveru neúspěšný, další objekt CertStore se použije k identifikaci serveru CRL a proces se opakuje od kroku 2.

   Pokud se jednalo o poslední CertStore v kolekci nebo pokud kolekce neobsahuje žádné objekty CertStore , proces vyhledávání se nezdařil a požadavek na připojení se nezdaří s kódem příčiny MQRC_SSL_CERT_STORE_ERROR.

Objekt Collection určuje pořadí, ve kterém se používají úložiště CertStores .

Pokud vaše aplikace používá metodu setSSLCertStores () k nastavení kolekce objektů CertStore , nelze již objekt MQConnectionFactory vázat do oboru názvů rozhraní JNDI. Pokus o provedení této operace způsobí výjimku. Není-li vlastnost sslCertStores nastavena, neprovádí se u certifikátu poskytnutého správcem front žádná kontrola odvolání. Tato vlastnost je ignorována, pokud není nastavena žádná sada CipherSuite .

Vlastnost objektu SSLRESETCOUNT
Tato vlastnost představuje celkový počet bajtů odeslaných a přijatých připojením, než bude znovu vyjednán tajný klíč použitý pro šifrování.

Počet odeslaných bajtů je číslo před šifrováním a počet přijatých bajtů je číslo po dešifrování. Počet bajtů také zahrnuje 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 tajným klíčem, který je znovu vyjednán po toku 4 MB dat, zadejte pro správce JMSAdmin následující příkaz:

```
ALTER CF(my.cf) SSLRESETCOUNT(4194304)
```

Aplikace může tuto vlastnost nastavit voláním metody setSSLResetCount () objektu ConnectionFactory .

Je-li hodnota této vlastnosti nula, což je výchozí hodnota, tajný klíč se nikdy znovu nevyjednává. Vlastnost je ignorována, pokud není nastavena žádná sada CipherSuite .

Vlastnost objektu SSLSocketFactory
Chcete-li upravit další aspekty připojení TLS pro aplikaci, vytvořte SSLSocketFactory a nakonfigurujte produkt JMS tak, aby jej používal.

Možná budete chtít upravit další aspekty připojení TLS pro aplikaci. Můžete například chtít inicializovat kryptografický hardware nebo změnit používané úložiště klíčů a úložiště údajů o důvěryhodnosti.
Za tímto účelem musí aplikace nejprve vytvořit objekt javax.net.ssl.SSLSocketFactory , který je odpovídajícím způsobem upraven. Informace o tom, jak to provést, naleznete v dokumentaci JSSE, protože přizpůsobitelné funkce se u jednotlivých poskytovatelů liší. Po získání vhodného objektu SSLSocketFactory použijte metodu MQConnectionFactory.setSSLSocketFactory () ke konfiguraci JMS pro použití přizpůsobeného objektu SSLSocketFactory .

Pokud vaše aplikace používá metodu setSSLSocketFactory () k nastavení přizpůsobeného objektu SSLSocketFactory , objekt MQConnectionFactory již nemůže být svázán s oborem názvů rozhraní JNDI. Pokus o provedení této operace způsobí výjimku. Není-li tato vlastnost nastavena, použije se výchozí objekt SSLSocketFactory . Podrobné informace o chování výchozího objektu SSLSocketFactory naleznete v dokumentaci JSSE. Tato vlastnost je ignorována, pokud není nastavena žádná sada CipherSuite .

Důležité: Nepředpokládejte, že použití vlastností SSL zajišťuje zabezpečení při načtení objektu ConnectionFactory z oboru názvů JNDI, který není sám o sobě zabezpečený. Konkrétně, standardní implementace LDAP rozhraní JNDI není zabezpečená. Útočník může napodobit server LDAP a uvést

aplikaci JMS v omyl, aby se připojila k chybnému serveru, aniž by si toho všiml. S vhodnými bezpečnostními opatřeními jsou zabezpečeny další implementace rozhraní JNDI (například implementace fscontext).

Provádění změn úložiště údajů o důvěryhodnosti nebo úložiště klíčů JSSE
Pokud provedete změny úložiště klíčů nebo úložiště údajů o důvěryhodnosti, musíte provést určité akce, aby byly změny vyzvednuty.

Pokud změníte obsah úložiště klíčů nebo úložiště údajů o důvěryhodnosti JSSE nebo změníte umístění souboru úložiště klíčů nebo úložiště údajů o důvěryhodnosti, aplikace IBM MQ classes for JMS , které jsou spuštěny v daném okamžiku, změny automaticky nevyzvednou. Aby se změny projevily, musí být provedeny následující akce:

- Aplikace musí zavřít všechna svá připojení a zničit všechna 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 tom, jak navrhujete aplikace, a na funkci, kterou poskytuje poskytovatel JSSE, může být možné provést tyto akce bez zastavení a restartování aplikací. Nejjednodušším řešením však může být zastavení a restartování aplikací.

TLS CipherSpecs a CipherSuites v adresáři IBM MQ classes for JMS
Schopnost aplikací IBM MQ classes for JMS navázat připojení ke správci front závisí na specifikaci CipherSpec zadané na konci serveru kanálu MQI a na sadě CipherSuite zadané na konci klienta.

V následující tabulce jsou uvedeny specifikace CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites.

Deprecated Měli byste si prostudovat téma Zamítnuté CipherSpecs , abyste zjistili, zda některá ze specifikací CipherSpecsuvedených v následující tabulce nebyla zamítnuta produktem IBM MQ , a pokud ano, při které aktualizaci specifikace CipherSpec byla zamítnuta.

Důležité: V seznamu CipherSuites jsou uvedeny ty, které jsou podporovány prostředím JRE (IBM Java Runtime Environment) dodávaným s produktem IBM MQ. Mezi uvedené CipherSuites patří ty, které jsou podporovány prostředím Oracle Java JRE. Další informace o konfiguraci aplikace pro použití prostředí Oracle Java JRE naleznete v tématu Konfigurace aplikace pro použití IBM Java nebo Oracle Java CipherSuite.

Tabulka také označuje protokol, který se používá pro komunikaci, a zda je či není sada CipherSuite v souladu se standardem FIPS 140-2.

Poznámka: V systému AIX, Linux, and Windowsposkytuje produkt IBM MQ kompatibilitu se standardem FIPS 140-2 prostřednictvím šifrovacího modulu IBM Crypto for C (ICC) . Certifikát pro tento modul byl přesunut do historického stavu. Zákazníci by si měli prohlédnout IBM Crypto for C (ICC) certifikát a měli by si být vědomi všech doporučení poskytnutých NIST. Náhradní modul FIPS 140-3 momentálně probíhá a jeho stav lze zobrazit jeho vyhledáním v modulech NIST CMVP v seznamu procesů.

Šifrovací sady označené jako vyhovující FIPS 140-2 lze použít, pokud nebyla aplikace nakonfigurována tak, aby vynucovala kompatibilitu se standardem FIPS 140-2, ale pokud byla pro aplikaci konfigurována shoda se standardem FIPS 140-2 (viz následující poznámky o konfiguraci), lze konfigurovat pouze ty CipherSuites , které jsou označeny jako kompatibilní se standardem FIPS 140-2; při pokusu o použití jiných CipherSuites dojde k chybě.

Poznámka: Každé prostředí JRE může mít více poskytovatelů kryptografického zabezpečení, z nichž každý může přispívat implementací stejné CipherSuite. Avšak ne všichni poskytovatelé zabezpečení mají certifikaci FIPS 140-2. Není-li pro aplikaci vynucena shoda se standardem FIPS 140-2, je možné, že bude použita necertifikovaná implementace CipherSuite . Necertifikované implementace nemusí fungovat v souladu se standardem FIPS 140-2, i když sada CipherSuite teoreticky splňuje minimální úroveň zabezpečení vyžadovanou standardem. Další informace o konfiguraci vynucení FIPS 140-2 v aplikacích IBM MQ JMS naleznete v následujících poznámkách.

Další informace o kompatibilních standardech FIPS 140-2 a Suite-B pro CipherSpecs a CipherSuitesnaleznete v tématu Určení CipherSpecs. Možná budete muset mít na paměti informace, které se týkají Federálních standardů zpracování informací.

Chcete-li použít úplnou sadu CipherSuites a pracovat s certifikovaným standardem FIPS 140-2 a/nebo Suite-B, je vyžadováno vhodné prostředí JRE. IBM Java 7 Service Refresh 4 Fix Pack 2 nebo vyšší úroveň prostředí IBM JRE poskytuje odpovídající podporu pro TLS 1.2 CipherSuites uvedené v části Tabulka 41 na stránce 250.

Chcete-li používat šifry TLS 1.3 , musí prostředí JRE spouštějící vaši aplikaci podporovat protokol TLS 1.3.

Poznámka: Chcete-li použít některé CipherSuites, je třeba v prostředí JRE nakonfigurovat soubory zásad 'bez omezení'. Další podrobnosti o nastavení souborů zásad v sadě SDK nebo JRE naleznete v tématu *IBM Soubory zásad sady SDK* v příručce *Security Reference for IBM SDK, Java Technology Edition* pro používanou verzi.

Tabulka 41. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites

CipherSpec **"1" na stránce 269**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem 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	TLS 1.2	yes

Tabulka 41. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites (pokračování)

CipherSpec **"1" na stránce 269**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem 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	TLS 1.2	yes

Tabulka 41. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites (pokračování)

CipherSpec **"1" na stránce 269**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem 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	TLS 1.2	yes

Tabulka 41. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites (pokračování)

CipherSpec **"1" na stránce 269**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem 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	TLS 1.2	yes

Tabulka 41. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites (pokračování)

CipherSpec **"1" na stránce 269**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem 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	TLS 1.2	yes

Tabulka 41. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites (pokračování)

CipherSpec **"1" na stránce 269**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem FIPS 140-2**
ECDHE_ECDSA_NULL_SHA256	SSL_ECDHE_ECDSA_WITH_NULL_SHA	TLS_ECDHE_ECDSA_WITH_NULL_SHA	TLS 1.2	no
ECDHE_ECDSA_RC4_128_SHA256	SSL_ECDHE_ECDSA_WITH_RC4_128_SHA	TLS_ECDHE_ECDSA_WITH_RC4_128_SHA	TLS 1.2	no

Tabulka 41. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites (pokračování)

CipherSpec **"1" na stránce 269**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem 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	TLS 1.2	yes

Tabulka 41. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites (pokračování)

CipherSpec **"1" na stránce 269**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem 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	TLS 1.2	yes

Tabulka 41. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites (pokračování)

CipherSpec **"1" na stránce 269**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem 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	TLS 1.2	yes

Tabulka 41. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites (pokračování)				
CipherSpec **"1" na stránce 269**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem 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	TLS 1.2	yes

Tabulka 41. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites (pokračování)

CipherSpec **"1" na stránce 269**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem 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	TLS 1.2	yes
ECDHE_RSA_NULL_SHA256	SSL_ECDHE_RSA_WITH_NULL_SHA	TLS_ECDHE_RSA_WITH_NULL_SHA	TLS 1.2	no

Tabulka 41. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites (pokračování)

CipherSpec **“1” na stránce 269**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem FIPS 140-2**
ECDHE_RSA_RC4_128_SHA256	SSL_ECDHE_RSA_WITH_RC4_128_SHA	TLS_ECDHE_RSA_WITH_RC4_128_SHA	TLS 1.2	no
TLS_RSA_WITH_3DES_EDE_CBC_SHA “2” na stránce 269	SSL_RSA_WITH_3DES_EDE_CBC_SHA	TLS_RSA_WITH_3DES_EDE_CBC_SHA	TLS 1.0	ne “4” na stránce 269

Tabulka 41. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites (pokračování)

CipherSpec **"1" na stránce 269**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem FIPS 140-2**
TLS_RSA_WITH_AES_128_CBC_SHA	SSL_RSA_WITH_AES_128_CBC_SHA	TLS_RSA_WITH_AES_128_CBC_SHA	TLS 1.0	ne "4" na stránce 269
TLS_RSA_WITH_AES_128_CBC_SHA256	SSL_RSA_WITH_AES_128_CBC_SHA256	TLS_RSA_WITH_AES_128_CBC_SHA256	TLS 1.2	ne "4" na stránce 269

Tabulka 41. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites (pokračování)

CipherSpec **“1” na stránce 269**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem FIPS 140-2**
TLS_RSA_WITH_AES_128_GCM_SHA256	SSL_RSA_WITH_AES_128_GCM_SHA256	TLS_RSA_WITH_AES_128_GCM_SHA256	TLS 1.2	ne “4” na stránce 269
TLS_RSA_WITH_AES_256_CBC_SHA	SSL_RSA_WITH_AES_256_CBC_SHA	TLS_RSA_WITH_AES_256_CBC_SHA	TLS 1.0	ne “4” na stránce 269

Tabulka 41. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites (pokračování)

CipherSpec **"1" na stránce 269**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem FIPS 140-2**
TLS_RSA_WITH_AES_256_CBC_SHA256	SSL_RSA_WITH_AES_256_CBC_SHA256	TLS_RSA_WITH_AES_256_CBC_SHA256	TLS 1.2	ne "4" na stránce 269
TLS_RSA_WITH_AES_256_GCM_SHA384	SSL_RSA_WITH_AES_256_GCM_SHA384	TLS_RSA_WITH_AES_256_GCM_SHA384	TLS 1.2	ne "4" na stránce 269

Tabulka 41. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites (pokračování)

CipherSpec **"1" na stránce 269**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem FIPS 140-2**
TLS_RSA_WITH_DES_CBC_SHA	SSL_RSA_WITH_DES_CBC_SHA	SSL_RSA_WITH_DES_CBC_SHA	TLS 1.0	no
TLS_RSA_WITH_NULL_SHA256	SSL_RSA_WITH_NULL_SHA256	TLS_RSA_WITH_NULL_SHA256	TLS 1.2	no

Tabulka 41. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites (pokračování)

CipherSpec **"1" na stránce 269**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem FIPS 140-2**
TLS_RSA_WITH_RC4_128_SHA256	SSL_RSA_WITH_RC4_128_SHA	SSL_RSA_WITH_RC4_128_SHA	TLS 1.2	no
ANY_TLS12	*TLS12	*TLS12	TLS 1.2	yes
TLS_AES_128_GCM_SHA256 "3" na stránce 269	TLS_AES_128_GCM_SHA256	TLS_AES_128_GCM_SHA256	TLS v1.3	no

Tabulka 41. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites (pokračování)

CipherSpec **“1” na stránce 269**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem FIPS 140-2**
TLS_AES_256_GCM_SHA384 “3” na stránce 269	TLS_AES_256_GCM_SHA384	TLS_AES_256_GCM_SHA384	TLS v1.3	no
TLS_CHACHA20_POLY1305_SHA256 “3” na stránce 269	TLS_CHACHA20_POLY1305_SHA256	TLS_CHACHA20_POLY1305_SHA256	TLS v1.3	no

Tabulka 41. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites (pokračování)

CipherSpec **"1" na stránce 269**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem FIPS 140-2**
TLS_AES_128_CCM_SHA256 "3" na stránce 269	TLS_AES_128_CCM_SHA256	TLS_AES_128_CCM_SHA256	TLS v1.3	no
TLS_AES_128_CCM_8_SHA256 "3" na stránce 269	TLS_AES_128_CCM_8_SHA256	TLS_AES_128_CCM_8_SHA256	TLS v1.3	no
Libovolný "3" na stránce 269	*ANY	*ANY	Více	no
ANY_TLS13 "3" na stránce 269	*TLS13	*TLS13	TLS V13	no

Tabulka 41. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites (pokračování)				
CipherSpec **"1" na stránce 269**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem FIPS 140-2**
ANY_TLS12_OR_HIGHER "3" na stránce 269	*TLS12ORHIGHER	*TLS12ORHIGHER	TLS 1.2 a vyšší	no
ANY_TLS13_OR_HIGHER "3" na stránce 269	*TLS13ORHIGHER	*TLS13ORHIGHER	TLS 1.3 a vyšší	no

Notes:

1. Toto je hodnota nakonfigurovaná na kanálu v produktu IBM MQ, včetně CCDT (binární nebo JSON).
2. Deprecated CipherSpec TLS_RSA_WITH_3DES_EDE_CBC_SHA je zamítnuta. Přesto jej lze použít k přenosu až 32 GB dat před ukončením připojení s chybou AMQ9288. Chcete-li se vyhnout této chybě, musíte se buď vyhnout použití trojitého DES, nebo povolit reset tajného klíče při použití této CipherSpec.
3. Chcete-li používat šifry TLS v1.3 , musí prostředí Java runtime environment (JRE), které spouští vaši aplikaci, podporovat protokol TLS v1.3.
4. V 9.3.0.17 V 9.3.5.1 V systémech IBM MQ 9.3.5 CSU 1 a IBM MQ 9.3.0 CSU 17prostředí JRE produktu IBM Java 8 odebere podporu výměny klíčů RSA při práci v režimu FIPS.

Konfigurace šifrovacích sad a shody se standardem FIPS v aplikaci IBM MQ classes for JMS

- Aplikace, která používá produkt IBM MQ classes for JMS , může k nastavení CipherSuite pro připojení použít jednu ze dvou metod:

- Vyvolejte metodu setSSLCipherSuite objektu ConnectionFactory .
- Pomocí nástroje pro administraci produktu IBM MQ JMS nastavte vlastnost SSLCIPHERSUITE objektu ConnectionFactory .

- Aplikace, která používá produkt IBM MQ classes for JMS , může k vynucení shody se standardem FIPS 140-2 použít jednu ze dvou metod:
  - Zavolejte metodu setSSLFipsRequired objektu ConnectionFactory .
  - Pomocí administračního nástroje produktu IBM MQ JMS nastavte vlastnost SSLFIPSREQUIRED objektu ConnectionFactory .

Konfigurace aplikace pro použití mapování produktu IBM Java nebo Oracle Java CipherSuite

Poznámka: V 9.3.3 V případě produktu Continuous Delivery from IBM MQ 9.3.3je Java Vlastnost systému `com.ibm.mq.cfg.useIBMCipherMappings`, která řídí, která mapování se používají, odebrána z produktu. V produktu IBM MQ 9.3.3lze šifrovací modul definovat buď jako název CipherSpec , nebo jako název CipherSuite a produkt IBM MQjej správně zpracuje. Následující tři soubory JAR Jackson jsou nezbytné, pokud vaše aplikace IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging vytvoří zabezpečená připojení TLS ke správci front:

- `jackson-annotations.jar`
- `jackson-core.jar`
- `jackson-databind.jar`

Důležité: Následující informace o souboru `com.ibm.mq.cfg.useIBMCipherMappings` platí pouze pro položky Long Term Support a Continuous Delivery před IBM MQ 9.3.3 .

Můžete nakonfigurovat, zda vaše aplikace používá výchozí mapování IBM Java CipherSuite na IBM MQ CipherSpec , nebo mapování Oracle CipherSuite na IBM MQ CipherSpec . Proto můžete použít TLS CipherSuites bez ohledu na to, zda vaše aplikace používá prostředí IBM JRE nebo Oracle JRE. Java Vlastnost systému `com.ibm.mq.cfg.useIBMCipherMappings` řídí, která mapování se použijí. Vlastnost může mít jednu z následujících hodnot:

ano
Použijte mapování IBM Java CipherSuite na IBM MQ CipherSpec .

Tato hodnota je výchozí hodnota.

ne
Použijte mapování Oracle CipherSuite na IBM MQ CipherSpec .

Další informace o použití šifer IBM MQ Java a TLS viz příspěvek blogu MQdev MQ Java, šifry TLS, jiné nežIBM JRE a APAR IT06775, IV66840, IT09423, IT10837.

Omezení interoperability

Určité CipherSuites mohou být kompatibilní s více než jednou IBM MQ CipherSpec, v závislosti na používaném protokolu. Podporována je však pouze kombinace CipherSuite/CipherSpec , která používá verzi TLS uvedenou v tabulce 1. Při pokusu o použití nepodporovaných kombinací CipherSuites a CipherSpecs dojde k selhání s příslušnou výjimkou. Instalace používající libovolnou z těchto kombinací CipherSuite/CipherSpec by se měly přesunout na podporovanou kombinaci.

Následující tabulka zobrazuje CipherSuites , na které se toto omezení vztahuje.

Tabulka 42. CipherSuites a jejich podporované a nepodporované CipherSpecs		
CipherSuite	**Podporovaná specifikace TLS CipherSpec**	**Nepodporovaná specifikace SSL CipherSpec**
SSL_RSA_WITH_3DES_EDE_CBC_SHA	TLS_RSA_WITH_3DES_EDE_CBC_SHA "1" na stránce 271	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. Deprecated Tato CipherSpec TLS_RSA_WITH_3DES_EDE_CBC_SHA je zamítnutá. Přesto jej lze použít k přenosu až 32 GB dat před ukončením připojení s chybou AMQ9288. Chcete-li se vyhnout této chybě, musíte se buď vyhnout použití trojitého DES, nebo povolit reset tajného klíče při použití této CipherSpec.

Zápis uživatelských procedur kanálu v Java pro IBM MQ classes for JMS
Uživatelské procedury kanálu vytvoříte definováním tříd Java , které implementují určená rozhraní.

Úvod k uživatelským procedurám zabezpečení naleznete v tématu Programy uživatelských procedur pro zabezpečení kanálu .

V balíku com.ibm.mq.exits jsou definována tři rozhraní:

- WMQSendExit, pro uživatelskou proceduru odeslání
- WMQReceiveExitpro uživatelskou proceduru příjmu.
- WMQSecurityExit, pro uživatelskou proceduru zabezpečení

Následující ukázkový kód 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 přijímá jako parametry objekt MQCXP a objekt MQCD. Tyto objekty představují struktury MQCXP a MQCD definované v procedurálním rozhraní.

Při volání uživatelské procedury pro odesílání 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 svou hodnotu data, která mají být odeslána správci front serveru. Pokud však uživatelská procedura pro odesílání není poslední uživatelskou procedurou pro odesílání v posloupnosti uživatelských procedur pro odesílání, vrácená data se namísto toho předají další uživatelské proceduře pro odesílání v posloupnosti. Uživatelská procedura odeslání může vrátit upravenou verzi dat, která obdrží v parametru agentBuffer , nebo může vrátit data beze změny. Nejjednodušším možným výstupním tělesem je proto:

```
{ return agentBuffer; }
```

Při volání uživatelské procedury pro příjem obsahuje parametr agentBuffer data přijatá ze správce front serveru. Uživatelská procedura pro příjem vrací jako svou hodnotu data, která mají být předána aplikaci produktem IBM MQ classes for JMS. Pokud však uživatelská procedura pro příjem není poslední uživatelskou procedurou pro příjem v posloupnosti uživatelských procedur pro příjem, vrácená data se namísto toho předají další uživatelské proceduře pro příjem v posloupnosti.

Když je volána uživatelská procedura zabezpečení, parametr agentBuffer obsahuje data, která byla přijata v toku zabezpečení z uživatelské procedury zabezpečení na konci serveru připojení. Uživatelská procedura zabezpečení vrací jako svou hodnotu data, která mají být odeslána v toku zabezpečení do uživatelské procedury zabezpečení serveru.

Uživatelské procedury kanálu jsou volány s vyrovnávací pamětí, která má záložní pole. Pro dosažení nejlepšího výkonu by měla uživatelská procedura vrátit vyrovnávací paměť s podpůrným polem.

Uživatelské proceduře kanálu lze při volání předat až 32 znaků uživatelských dat. Uživatelská procedura přistupuje k uživatelským datům voláním metody getExitData () objektu MQCXP. Přestože uživatelská procedura může změnit uživatelská data voláním metody setExitData (), uživatelská data se aktualizují při každém volání uživatelské procedury. Veškeré změny provedené v uživatelských datech jsou proto ztraceny. Uživatelská procedura však může předávat data z jednoho volání do druhého pomocí uživatelské oblasti uživatelské procedury objektu MQCXP. Uživatelská procedura přistupuje k uživatelské oblasti uživatelské procedury pomocí odkazu voláním metody getExitUserArea().

Každá třída ukončení musí mít konstruktor. Konstruktor může být buď výchozí konstruktor, jak ukazuje předchozí příklad, nebo konstruktor s řetězcovým parametrem. Konstruktor je volán, aby vytvořil instanci třídy ukončení pro každou uživatelskou proceduru definovanou ve třídě. Proto je v předchozím příkladu vytvořena instance třídy MyMQExits pro uživatelskou proceduru odeslání, pro uživatelskou proceduru příjmu 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 výchozí konstruktor i konstruktor s jedním parametrem, má přednost konstruktor s jedním parametrem.

Nezavírejte připojení z uživatelské procedury kanálu.

Když jsou data odeslána na konec připojení serveru, šifrování TLS se provede *po* volání všech uživatelských procedur kanálu. Podobně, když jsou data přijata z konce připojení serveru, dešifrování TLS se provede *před* voláním všech uživatelských procedur kanálu.

Ve verzích IBM MQ classes for JMS starších než IBM WebSphere MQ 7.0byly uživatelské procedury kanálu implementovány pomocí rozhraní MQSendExit, MQReceiveExita MQSecurityExit. Tato rozhraní můžete stále používat, ale nová rozhraní jsou upřednostňována pro lepší funkci a výkon.

Konfigurace IBM MQ classes for JMS pro použití uživatelských procedur kanálu
Aplikace IBM MQ classes for JMS může používat uživatelské procedury pro zabezpečení kanálu, odesílání a příjem v kanálu MQI, který se spustí při připojení aplikace ke správci front. Aplikace může používat uživatelské procedury napsané v adresáři Java, C nebo C + +. Aplikace může také použít posloupnost uživatelských procedur send nebo receive, které jsou spouštěny v posloupnosti.

Následující vlastnosti se používají k určení uživatelské procedury pro odeslání nebo posloupnosti uživatelských procedur pro odeslání, které používá připojení JMS :

- Vlastnost **SENDEXIT** objektu MQConnectionFactory .

- Vlastnost **sendexit** ve specifikaci aktivace používané adaptérem prostředků IBM MQ pro příchozí komunikaci.
- Vlastnost **sendexit** objektu ConnectionFactory používaná adaptérem prostředků IBM MQ pro výstupní komunikaci.

Hodnota vlastnosti je řetězec, který se skládá z jedné nebo více položek oddělených čárkami. Každá položka identifikuje uživatelskou proceduru odeslání jedním z následujících způsobů:

- Název třídy, která implementuje rozhraní `WMQSendExit` pro uživatelskou proceduru odeslání zapsanou v souboru Java.
- Řetězec ve formátu *libraryName* (*entryPointName*) pro uživatelskou proceduru odeslání napsanou v jazyce C nebo C + +.

Podobně následující vlastnosti určují uživatelskou proceduru příjmu nebo posloupnost uživatelských procedur příjmu používaných připojením:

- Vlastnost **RECEXIT** objektu MQConnectionFactory .
- Vlastnost **receiveexit** ve specifikaci aktivace používané adaptérem prostředků IBM MQ pro příchozí komunikaci.
- Vlastnost **receiveexit** objektu ConnectionFactory používaná adaptérem prostředků IBM MQ pro výstupní komunikaci.

Následující vlastnosti určují uživatelskou proceduru zabezpečení používanou připojením:

- Vlastnost **SECEXIT** objektu MQConnectionFactory .
- Vlastnost **securityexit** ve specifikaci aktivace používané adaptérem prostředků IBM MQ pro příchozí komunikaci.
- Vlastnost **securityexit** objektu ConnectionFactory používaná adaptérem prostředků IBM MQ pro výstupní komunikaci.

Pro 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 načteny vlastním zavaděčem tříd. Chcete-li najít uživatelskou proceduru kanálu, zavaděč tříd vyhledá následující umístění v zadaném pořadí.

1. Cesta ke třídám určená vlastností **com.ibm.mq.cfg.ClientExitPath.JavaExitsClasspath** nebo atributem **JavaExitsClassPath** v sekci Kanály konfiguračního souboru klienta IBM MQ .
2. Deprecated Cesta ke třídám určená Java systémovou vlastností **com.ibm.mq.exitClasspath**. Všimněte si, že tato vlastnost je nyní zamítnuta.
3. IBM MQ ukončí adresář, jak ukazuje Tabulka 43 na stránce 273. Zavaděč tříd nejprve vyhledá v adresáři soubory tříd, které nejsou zabaleny v souborech archivu Java (JAR). Není-li uživatelská procedura kanálu nalezena, zavaděč tříd poté prohledá soubory JAR v adresáři.

Tabulka 43. Adresář ukončení IBM MQ

Platforma	**Adresář**
Linux AIX AIX and Linux	`/var/mqm/exits` (32bitové uživatelské procedury kanálu) `/var/mqm/exits64` (64bitové uživatelské procedury kanálu)
Windows Windows	*instalační_datový_adresář*\exits kde *instal_data_dir* je adresář, který jste vybrali pro datové soubory IBM MQ během instalace. Výchozí adresář je C:\ProgramData\IBM\MQ.

Poznámka: Pokud uživatelská procedura kanálu existuje ve více než jednom umístění, konzola IBM MQ classes for JMS načte první nalezenou instanci.

Nadřízeným objektem zavaděče tříd je zavaděč tříd, který se používá k načtení souboru IBM MQ classes for JMS. Proto je možné, aby nadřízený zavaděč tříd zavedl uživatelskou proceduru kanálu, pokud ji nelze nalézt v žádném z předchozích umístění. Pokud však používáte agenta IBM MQ classes for JMS v prostředí, jako je aplikační server JEE , pravděpodobně nebudete moci ovlivnit výběr zavaděče nadřízené třídy, a proto by měl být zavaděč tříd konfigurován nastavením Java systémové vlastnosti **com.ibm.mq.cfg.ClientExitPath.JavaExitsClasspath** na aplikačním serveru.

Pokud je vaše aplikace spuštěna s povoleným produktem Java security manager , musí mít konfigurační soubor zásad používaný běhovým prostředím produktu Java , ve kterém je aplikace spuštěna, oprávnění k načtení třídy uživatelské procedury kanálu. Informace o tom, jak to provést, naleznete v tématu Spuštění tříd IBM MQ pro aplikace JMS v části Java Správce zabezpečení.

Rozhraní `MQSendExit`, `MQReceiveExita` `MQSecurityExit` dodávaná s verzemi staršími než IBM WebSphere MQ 7.0 jsou stále podporována. Pokud používáte uživatelské procedury kanálu, které implementují tato rozhraní, musí být v cestě ke třídám přítomen parametr `com.ibm.mq.jar` .

Informace o způsobu zápisu uživatelských procedur kanálu v jazyce C naleznete v části "Programy uživatelské procedury kanálu pro kanály systému zpráv" na stránce 928. Uživatelské programy kanálu napsané v jazyce C nebo C++ musíte uložit do adresáře zobrazeného v souboru Tabulka 43 na stránce 273.

Pokud vaše aplikace používá tabulku CCDT (Client Channel Definition Table) pro připojení ke správci front, postupujte podle části "Použití tabulky definic kanálů klienta s volbou IBM MQ classes for JMS" na stránce 275.

Určení uživatelských dat, která mají být předána uživatelským procedur kanálu při použití IBM MQ classes for JMS
Uživatelské proceduře kanálu lze při volání předat až 32 znaků uživatelských dat.

Vlastnost SENDEXITINIT objektu MQConnectionFactory určuje uživatelská data, která jsou při volání předána každé uživatelské proceduře pro odeslání. Hodnota vlastnosti je řetězec, který se skládá z jedné nebo více položek uživatelských dat oddělených čárkami. Pozice každé položky uživatelských dat v řetězci určuje, které uživatelské proceduře odeslání se v posloupnosti uživatelských procedur předávají uživatelská data. Například první položka uživatelských dat v řetězci je předána první uživatelské proceduře odeslání v posloupnosti uživatelských procedur odeslání.

Vlastnost SENDEXITINIT můžete nastavit pomocí nástroje pro administraci IBM MQ JMS nebo IBM MQ Explorer. Alternativně může aplikace nastavit vlastnost voláním metody setSendExitInit().

Podobně vlastnost RECEXITINIT objektu ConnectionFactory určuje uživatelská data předávaná každé uživatelské proceduře pro příjem a vlastnost SECEXITINIT určuje uživatelská data předaná uživatelské proceduře pro zabezpečení. Tyto vlastnosti můžete nastavit pomocí nástroje pro administraci produktu IBM MQ JMS nebo pomocí nástroje IBM MQ Explorer. Alternativně může aplikace nastavit vlastnosti voláním metod setReceiveExitInit() a setSecurityExitInit().

Při zadávání uživatelských dat předávaných uživatelským procedur kanálu mějte na paměti následující pravidla:

- Pokud je počet položek uživatelských dat v řetězci větší než počet uživatelských procedur v posloupnosti, přebytečné položky uživatelských dat se ignorují.
- Je-li počet položek uživatelských dat v řetězci menší než počet uživatelských procedur v posloupnosti, je každá nespecifikovaná položka uživatelských dat nastavena na prázdný řetězec. Dvě čárky v posloupnosti v řetězci nebo čárka na začátku řetězce také označují nespecifikovanou položku uživatelských dat.

Pokud aplikace používá tabulku CCDT (Client Channel Definition Table) pro připojení ke správci front, budou při volání předána uživatelská data určená v definici kanálu připojení klienta uživatelským procedur kanálu. Další informace o použití tabulky definic kanálů klienta viz "Použití tabulky definic kanálů klienta s volbou IBM MQ classes for JMS" na stránce 275.

Použití tabulky definic kanálů klienta s volbou IBM MQ classes for JMS
Aplikace IBM MQ classes for JMS může používat definice kanálů připojení klienta, které jsou uloženy v tabulce CCDT (Client Channel Definition Table). Konfigurujete objekt ConnectionFactory pro použití tabulky CCDT. Existují určitá omezení týkající se jeho použití.

Jako alternativu k vytvoření definice kanálu připojení klienta nastavením určitých vlastností objektu ConnectionFactory může aplikace IBM MQ classes for JMS používat definice kanálů připojení klienta, které jsou uloženy v tabulce definic kanálů klienta. Tyto definice jsou vytvářeny příkazy jazyka MQSC (IBM MQ Script) nebo PCF (IBM MQ Programmable Command Format). Když aplikace vytvoří objekt připojení, produkt IBM MQ classes for JMS vyhledá v tabulce definic kanálů klienta vhodnou definici kanálu připojení klienta a použije tuto definici kanálu ke spuštění kanálu MQI. Další informace o definičních tabulkách kanálů klienta a o způsobu jejich konstrukce naleznete v tématu Tabulka definic kanálů klienta.

Chcete-li použít tabulku definic kanálů klienta, musí být vlastnost CCDTURL objektu ConnectionFactory nastavena na objekt URL . Produkt IBM MQ classes for JMS nečte informace o tabulce CCDT z konfiguračního souboru IBM MQ MQI client , i když se z něj používají některé další hodnoty (viz “Konfigurační soubor IBM MQ classes for JMS/Jakarta Messaging” na stránce 95 , pro které hodnota platí). Objekt URL zapouzdřuje jednotný lokátor prostředků (URL), který 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 může aplikace nastavit vlastnost vytvořením objektu URL a voláním metody setCCDTURL() objektu ConnectionFactory .

Pokud například soubor ccdt1.tab obsahuje tabulku definic kanálů klienta a je uložen ve stejném systému, ve kterém 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 systému, na kterém je aplikace spuštěna. Pokud lze k souboru přistupovat 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 stejné hodnoty, které lze použít pro parametr **QMgrName** ve volání MQCONN vydaném klientskou aplikací, která používá rozhraní MQI (Message Queue Interface). Další informace o významu těchto hodnot viz MQCONN. Vlastnost QMANAGER můžete nastavit pomocí nástroje pro administraci IBM MQ JMS nebo Průzkumníka IBM MQ . Volitelně 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 určené vlastností CCDTURL, použije vlastnost QMANAGER k vyhledání vhodné definice kanálu připojení klienta a poté použije definici kanálu ke spuštění kanálu MQI pro správce front.

Mějte na zřeteli, že vlastnosti CCDTURL a CHANNEL objektu ConnectionFactory nemohou být nastaveny současně, když aplikace volá metodu createConnection(). Jsou-li nastaveny obě vlastnosti, metoda vygeneruje výjimku. Vlastnost CCDTURL nebo CHANNEL se považuje za nastavenou, pokud má jinou hodnotu než null, prázdný řetězec nebo řetězec obsahující všechny prázdné znaky.

Pokud produkt IBM MQ classes for JMS najde v tabulce definic kanálů klienta vhodnou definici kanálu připojení klienta, použije ke spuštění kanálu MQI pouze informace extrahované z tabulky. Všechny vlastnosti související s kanálem objektu ConnectionFactory jsou ignorovány.

Všimněte si zejména následujících bodů, pokud používáte TLS:

- Kanál MQI používá protokol TLS pouze v případě, že definice kanálu extrahovaná z tabulky definic kanálů klienta určuje název CipherSpec podporované produktem IBM MQ classes for JMS.
- Tabulka definic kanálů klienta obsahuje také informace o umístění serverů LDAP (Lightweight Directory Access Protocol), které obsahují seznamy odvolaných certifikátů (CRL). Produkt IBM MQ classes for JMS používá pouze tyto informace pro přístup k serverům LDAP, které obsahují seznamy CRL.
- Tabulka definic kanálů 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. Protokol OCSP však můžete konfigurovat podle popisu v části Protokol OCSP (Online Certificate Status Protocol) v produktu Java a JMS klientské aplikace.

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 napsány v souboru Java. To například znamená, že parametr SCYEXIT v příkazu DEFINE CHANNEL pro vytvoření definice kanálu připojení klienta může určovat název třídy, která implementuje rozhraní WMQSecurityExit . Podobně může parametr SENDEXIT určovat název třídy, která implementuje rozhraní WMQSendExit , a parametr RCVEXIT může určovat název třídy, která implementuje rozhraní WMQReceiveExit . Další informace o zápisu uživatelské procedury kanálu v adresáři Javanaleznete v části"Zápis uživatelských procedur kanálu v Java pro IBM MQ classes for JMS" na stránce 271.

  Je také podporováno použití uživatelských procedur kanálu napsaných v jiném jazyce než v jazyce Java . Informace o tom, jak zadat parametry SCYEXIT, SENDEXIT a RCVEXIT v příkazu DEFINE CHANNEL pro uživatelské procedury kanálu zapsané v jiném jazyce, viz DEFINE CHANNEL.

Automatické opětovné připojení klienta JMS
Nakonfigurujte klienta JMS tak, aby se automaticky znovu připojoval po selhání sítě, správce front nebo serveru.

Normally, if a stand-alone IBM MQ classes for JMS application is connected to a queue manager by using the client transport, and the queue manager becomes unavailable for some reason (due to a network outage, a queue manager failure, or the queue manager being stopped, for example), the IBM MQ classes for JMS will throw a JMSException the next time the application tries to communicate with the queue manager. Aplikace musí zachytit výjimku JMSException a pokusit se znovu připojit ke správci front. Návrh aplikace můžete zjednodušit povolením automatického opětovného 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 opětovné připojení.

Použití této implementace automatického opětovného připojení klienta není v rámci aplikačních serverů Java Platform, Enterprise Edition podporováno. Alternativní implementace viz "Použití automatického opětovného připojení klienta v prostředí Java EE" na stránce 282 .

Použití automatického opětovného připojení klienta JMS
Pokud samostatná aplikace IBM MQ classes for JMS používá továrnu připojení, která má nastavenu vlastnost CONNECTIONNAMELIST nebo CCDTURL, je aplikace způsobilá používat automatické opětovné připojení klienta.

Automatické opětovné připojení klienta lze použít k opětovnému připojení ke správcům front včetně těch, kteří jsou součástí konfigurace vysoké dostupnosti (HA). Konfigurace HA zahrnují správce front s více instancemi, správce front RDQM nebo správce front HA na zařízení IBM MQ .

Chování funkce automatického opětovného připojení klienta, kterou poskytuje produkt IBM MQ classes for JMS , závisí na následujících vlastnostech:

JMS DOPRAVA vlastnosti továrny připojení (zkrácený název TRAN)

Volba TRANSPORT určuje, jak se aplikace používající továrnu připojení připojují ke správci front. Tato vlastnost musí být nastavena na hodnotu CLIENT, aby bylo možné použít automatické opětovné

připojení klienta. Automatické opětovné připojení klienta není k dispozici pro aplikace, které se připojují ke správci front, který používá továrnu připojení s vlastností TRANSPORT nastavenou na BIND, DIRECT nebo DIRECTHTTP.

JMS Vlastnost továrny připojení QMANAGER (Krátký název QMGR)

Vlastnost QMANAGER určuje název správce front, ke kterému se připojuje továrna připojení.

JMS Vlastnost továrny připojení CONNECTIONNAMELIST (Krátký název CRHOSTS)

Vlastnost CONNECTIONNAMELIST je seznam oddělený čárkami, kde každá položka obsahuje informace o názvu hostitele a portu, které se mají použít pro připojení ke správci front uvedenému ve vlastnosti QMANAGER při použití přenosu CLIENT. Seznam má následující formát: název hostitele (port), název hostitele (port).

JMS Vlastnost továrny připojení CCDTURL (Krátký název CCDT)

Vlastnost CCDTURL ukazuje na tabulku definic kanálů klienta, kterou produkt IBM MQ classes for JMS používá při připojování ke správci front pomocí tabulky CCDT.

JMS Vlastnost továrny připojení CLIENTRECONNECTOPTIONS (zkrácený název CROPT)

Volba CLIENTRECONNECTOPTIONS určuje, zda se konzola IBM MQ classes for JMS pokusí o automatické připojení ke správci front jménem aplikace v případě, že bude správce front k dispozici.

Atribut DefRecon v sekci Kanály konfiguračního souboru klienta

Atribut DefRecon poskytuje volbu administrace, která umožňuje všem aplikacím automatické opětovné připojení nebo zakázání automatického opětovného připojení pro aplikace, které jsou zapsány pro automatické opětovné připojení.

Automatické opětovné připojení klienta je k dispozici pouze v případě, že se aplikace úspěšně připojí ke správci front.

Když se aplikace připojí ke správci front, který používá přenos CLIENT, produkt IBM MQ classes for JMS použije hodnotu vlastnosti továrny připojení CLIENTRECONNECTOPTIONS k určení, zda se má použít automatické opětovné připojení klienta, pokud se správce front, ke kterému je aplikace připojena, stane nedostupným. Tabulka 1 zobrazuje možné hodnoty pro vlastnost CLIENTRECONNECTOPTIONS a chování IBM MQ classes for JMS pro každou z těchto hodnot:

Tabulka 44. Možné hodnoty vlastností CLIENTRECCECTOPTIONS.

CLIENTRECONNECTOPTIONS	**Chování uživatele IBM MQ classes for JMS**
ANY	Je-li nastaven parametr CONNECTIONNAMELIST, použijte hodnotu vlastnosti CONNECTIONNAMELIST k otevření připojení ke kombinaci názvu hostitele a portu a připojení k libovolnému správci front. Chcete-li použít tuto volbu automatického opětovného připojení klienta, musí být vlastnost QMANAGER nastavena na výchozí hodnotu nebo na hodnotu "*". Je-li nastavena hodnota CCDTURL, otevřete tabulku definic kanálů klienta, která je určena vlastností CCDTURL, vyberte položku v tabulce a poté pomocí této položky spusťte kanál připojení klienta ke správci front. Chcete-li použít tuto volbu automatického opětovného připojení klienta, musí být vlastnost QMANAGER nastavena na jednu z následujících možností: • Hvězdička (*) • Hvězdička (*) následovaná názvem skupiny správců front • Prázdný řetězec nebo řetězec, který obsahuje všechny prázdné znaky
ASDEF	Pomocí hodnoty DefRecon určete, zda je k dispozici automatické opětovné připojení klienta.
VYPNUTO	Neprovádějte žádné automatické opětovné připojení klienta a nevracejte aplikaci výjimku JMSException.
QMGR	Určuje, že klient se musí znovu připojit ke stejnému správci front. Tato volba musí být použita pro řešení vysoké dostupnosti, kde je vyžadováno opětovné připojení k jiné instanci stejného správce front. Je-li nastaven parametr CONNECTIONNAMELIST, pomocí hodnoty vlastnosti CONNECTIONNAMELIST otevřete připojení ke kombinaci názvu hostitele a portu a připojte se ke správci front uvedenému ve vlastnosti QMANAGER. Je-li nastavena hodnota CCDTURL, otevřete tabulku definic kanálů klienta určenou vlastností CCDTURL, vyhledejte položky v tabulce, které odpovídají názvu správce front určenému vlastností QMANAGER, a poté pomocí těchto položek spusťte kanál připojení klienta k danému správci front.

Je-li parametr CONNECTIONNAMELIST nastaven, při provádění automatického opětovného připojení klienta produkt IBM MQ classes for JMS použije informace ve vlastnosti továrny připojení CONNNECTIONNAMELIST k určení, ke kterému systému se má znovu připojit.

Produkt IBM MQ classes for JMS se na počátku pokusí znovu připojit pomocí názvu hostitele a portu, který je uveden v první položce v parametru CONNECTIONNAMELIST. Je-li vytvořeno připojení, produkt

IBM MQ classes for JMS se poté pokusí připojit ke správci front, který má název uvedený ve vlastnosti QMANAGER. Pokud lze vytvořit připojení ke správci front, produkt IBM MQ classes for JMS znovu otevře všechny objekty IBM MQ , které aplikace otevřela před automatickým opětovným připojením klienta, a bude pokračovat v práci jako dříve.

Pokud nelze vytvořit připojení k požadovanému správci front pomocí první položky v parametru CONNECTIONNAMELIST, produkt IBM MQ classes for JMS se pokusí o druhou položku v parametru CONNECTIONNAMELIST atd.

Když produkt IBM MQ classes for JMS zkusí všechny položky v parametru CONNECTIONNAMELIST, počkají určitou dobu, než se znovu pokusí o připojení. Chcete-li provést nový pokus o opětovné připojení, IBM MQ classes for JMS se spustí s první položkou v parametru CONNECTIONNAMELIST. Pak zkusí každou položku v parametru CONNECTIONNAMELIST postupně, dokud nedojde k opětovnému připojení nebo dokud nebude dosaženo konce parametru CONNECTIONNAMELIST. V tomto okamžiku produkt IBM MQ classes for JMS čeká určitou dobu, než operaci zopakuje.

Je-li nastavena hodnota CCDTURL, produkt IBM MQ classes for JMS při provádění automatického opětovného připojení klienta použije 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, produkt IBM MQ classes for JMS se pokusí znovu připojit k požadovanému správci front pomocí této položky. Pokud lze vytvořit připojení ke správci front, produkt IBM MQ classes for JMS znovu otevře všechny objekty IBM MQ , které aplikace otevřela před automatickým opětovným připojením klienta, a bude pokračovat v práci jako dříve.

Pokud nelze vytvořit připojení k požadovanému správci front, produkt IBM MQ classes for JMS vyhledá v tabulce definic kanálů klienta jinou vhodnou položku a pokusí se ji použít atd.

Když agent IBM MQ classes for JMS zkusí všechny vhodné položky v tabulce definic kanálů klienta, počkají určitou dobu, než se znovu pokusí o připojení. Chcete-li provést nový pokus o opětovné připojení, produkt IBM MQ classes for JMS znovu analyzuje tabulku definic kanálů klienta a pokusí se o první vhodnou položku. Poté se pokusí o každou vhodnou položku v tabulce definic kanálů klienta postupně, dokud nedojde k opětovnému připojení nebo k pokusu o poslední vhodnou položku v tabulce definic kanálů klienta. V tomto okamžiku produkt IBM MQ classes for JMS před dalším pokusem počká určitou dobu.

Bez ohledu na to, zda používáte CONNECTIONNAMELIST nebo CCDTURL, proces automatického opětovného připojení klienta pokračuje, dokud se IBM MQ classes for JMS úspěšně znovu nepřipojí ke správci front uvedenému ve vlastnosti QMANAGER.

Standardně se pokusy o opětovné připojení opakují v následujících intervalech:

- První pokus se provede po počáteční prodlevě 1 sekunda plus náhodný prvek až do 250 milisekund.
- Druhý pokus se provede 2 sekundy plus náhodný interval až 500 milisekund po selhání prvního pokusu.
- Třetí pokus se provede 4 sekundy plus náhodný interval až 1 sekunda, po druhém pokusu selže.
- Čtvrtý pokus se provede 8 sekund plus náhodný interval až 2 sekundy po selhání třetího pokusu.
- Pátý pokus se provede 16 sekund plus náhodný interval až 4 sekundy po selhání čtvrtého pokusu.
- Šestý pokus a všechny následné pokusy jsou provedeny 25 sekund plus náhodný interval až 6 sekund a 250 milisekund po selhání předchozího pokusu.

Pokusy o opětovné připojení jsou zpožděny o intervaly, které jsou částečně pevné a částečně náhodné. To má zabránit všem aplikacím IBM MQ classes for JMS , které byly připojeny ke správci front, který již není k dispozici, v opakujícím se připojení současně.

Potřebujete-li zvýšit výchozí hodnoty tak, aby přesněji odrážely dobu potřebnou k zotavení správce front nebo aby se záložní správce front stal aktivním, upravte atribut ReconDelay v sekci Kanál konfiguračního souboru klienta. Další informace naleznete v části CHANNELS stanza konfiguračního souboru klienta.

To, zda aplikace IBM MQ classes for JMS bude i nadále správně fungovat i po automatickém opětovném připojení, závisí na jejím návrhu. Přečtěte si související témata, abyste porozuměli tomu, jak mohou aplikace používat funkci automatického opětovného připojení.

Kódy příčiny indikující, že správce front již není k dispozici
Jaké kódy příčiny označují, že správce front již není k dispozici nebo není dosažitelný při pokusu o automatické opětovné připojení produktu IBM MQ classes for JMS .

Produkt “Automatické opětovné připojení klienta JMS” na stránce 276 poskytuje přehled výjimek JMSExceptions a o tom, jak mohou být aplikace automaticky restartovány, a informace v části “Použití automatického opětovného připojení klienta JMS” na stránce 276 podrobně uvádějí požadavky na automatické opětovné připojení klienta.

Následující informace uvádí kódy příčiny IBM MQ , které by měla vaše aplikace zkontrolovat:

RC2009
MQRC_CONNECTION_BROKEN

RC2059
MQRC_Q_MGR_NOT_AVAILABLE

RC2161
MQRC_Q_MGR_QUIESCING

RC2162
MQRC_Q_MGR_ZASTAVENÍ

RC2202
MQRC_CONNECTION_QUIESCING

RC2203
MQRC_CONNECTION_ZASTAVENÍ

RC2223
MQRC_Q_MGR_NOT_ACTIVE

RC2279
MQRC_CHANNEL_STOPPED_BY_USER

RC2537
MQRC_CHANNEL_NOT_AVAILABLE

RC2538
MQRC_HOST_NENÍ k dispozici

Většina výjimek JMSExceptions vrácených podnikovým aplikacím obsahuje propojenou výjimku MQException, která obsahuje kód příčiny. Chcete-li implementovat logiku opakování pro kódy příčiny v předchozím seznamu, měly by podnikové aplikace zkontrolovat tuto propojenou výjimku pomocí kódu podobného následujícímu příkladu:

```
} catch (JMSException ex) {
        Exception linkedEx = ex.getLinkedException();
        if (ex.getLinkedException() != null) {
            if (linkedEx instanceof MQException) {
                MQException mqException = (MQException) linkedEx;
                int reasonCode = mqException.reasonCode;
                // Handle the reason code accordingly
            }
        }
    }
```

Související pojmy
IBM MQ třídy pro JMS

Použití automatického opětovného připojení klienta v prostředích Java SE a Java EE
Můžete použít automatické opětovné připojení klienta IBM MQ k usnadnění různých řešení vysoké dostupnosti (HA) a zotavení z havárie (DR) v prostředí Java SE a Java EE .

Různá řešení HA a DR jsou k dispozici na různých platformách:

- Multi Správci front s více instancemi jsou instance stejného správce front konfigurovaného na různých serverech (viz Správci front s více instancemi). Jedna instance správce front je definována jako

aktivní instance a jiná instance je definována jako rezervní instance. Pokud se aktivní instance nezdaří, správce front pro více instancí se automaticky restartuje na záložním serveru.

Aktivní i rezervní správci front mají stejný identifikátor správce front (QMID). Klientské aplikace IBM MQ , které se připojují ke správci front s více instancemi, lze konfigurovat tak, aby se automaticky znovu připojovaly k rezervní instanci správce front pomocí automatického opětovného připojení klienta.

- Linux RDQM (správce front replikovaných dat) je řešení vysoké dostupnosti, které je k dispozici na platformách Linux (viz Vysoká dostupnost RDQM). Konfigurace RDQM se skládá ze tří serverů nakonfigurovaných ve skupině s vysokou dostupností (HA), z nichž každý má instanci správce front. Jedna instance je spuštěný správce front, který synchronně replikuje svá data do dalších dvou instancí. Pokud server, na kterém je spuštěn tento správce front, selže, spustí se další instance správce front a bude mít aktuální data pro práci. Tři instance správce front sdílejí plovoucí adresu IP, takže klienti musí být konfigurováni pouze s jednou adresou IP. Klientské aplikace, které se připojují ke správci front RDQM, lze konfigurovat tak, aby se automaticky znovu připojovaly k rezervní instanci správce front pomocí automatického opětovného připojení klienta.
- MQ Appliance Řešení vysoké dostupnosti může být také poskytnuto dvojicí zařízení IBM MQ (viz Vysoká dostupnost a Zotavení z havárie v dokumentaci k produktu IBM MQ Appliance). Správce front HA je spuštěn na jednom ze zařízení, zatímco synchronně replikuje data do rezervní instance správce front na druhém zařízení. Pokud primární zařízení selže, správce front se automaticky spustí a spustí na druhém zařízení. Dvě instance správce front lze konfigurovat tak, aby sdílely plovoucí adresu IP, takže klienty je třeba konfigurovat pouze s jedinou adresou IP. Klientské aplikace, které se připojují ke správci front HA na zařízení IBM MQ , lze nakonfigurovat tak, aby se automaticky znovu připojovaly k instanci pohotovostního režimu správce front pomocí automatického opětovného připojení klienta.

Poznámka: V prostředích systému Java EE , jako je například prostředí WebSphere Application Server, není automatické opětovné připojení klienta se specifikacemi aktivace pomocí funkce poskytované produktem IBM MQ classes for JMS podporováno. Adaptér prostředků IBM MQ poskytuje vlastní mechanizmus pro opětovné připojení specifikací aktivace, pokud se správce front, ke kterému se připojovala specifikace aktivace, stane nedostupným. Další informace viz “Podpora automatického opětovného připojení klienta v prostředí Java EE” na stránce 283.

Související pojmy

Správci front s více instancemi

Automatické opětovné připojení klienta

Související odkazy

vysoká dostupnost rdqm

Použití automatického opětovného připojení klienta v prostředí Java SE

Aplikace používající IBM MQ classes for JMS spuštěné v prostředí Java SE mohou používat funkci automatického opětovného připojení klienta prostřednictvím vlastnosti továrny připojení **CLIENTRECONNECTOPTIONS**.

Vlastnost továrny připojení **CLIENTRECONNECTOPTIONS** používá dvě další vlastnosti továrny připojení **CONNECTIONNAMELIST** a **CCDTURL**k určení způsobu připojení k serveru, na kterém je spuštěn správce front.

CONNECTIONNAMELIST vlastnost

Vlastnost **CONNECTIONNAMELIST** je seznam oddělený čárkami, který obsahuje informace o názvu hostitele a portu, které se mají použít pro připojení ke správci front v režimu klienta. Tato vlastnost se používá s hodnotami **QMANAGER** a **CHANNEL** . Když aplikace používá vlastnost **CONNECTIONNAMELIST** k vytvoření připojení klienta, produkt IBM MQ classes for JMS se pokusí připojit ke každému hostiteli v pořadí seznamu. Není-li první hostitel správce front k dispozici, produkt IBM MQ classes for JMS se pokusí připojit k dalšímu hostiteli v seznamu. Pokud je dosaženo konce seznamu názvů připojení bez vytvoření připojení, IBM MQ classes for JMS vygenerujte kód příčiny `MQRC_QMGR_NOT_AVAILABLE` IBM MQ .

Pokud dojde k selhání správce front, ke kterému je aplikace připojena, všechny aplikace, které pro připojení k tomuto správci front používaly rozhraní **CONNECTIONNAMELIST** , obdrží výjimku informující o tom, že správce front není k dispozici. Aplikace musí zachytit výjimku a vymazat všechny prostředky, které používala. Chcete-li vytvořit připojení, musí aplikace použít továrnu připojení. Továrna připojení se znovu pokusí připojit ke každému hostiteli v pořadí seznamu. Správce front, který selhal, nyní není k dispozici. Továrna připojení se pokusí připojit k jinému hostiteli v seznamu.

CCDTURL vlastnost

Vlastnost **CCDTURL** obsahuje lokátor jednotného prostředku (URL), který ukazuje na tabulku CCDT (Client Channel Definition Table). Tato vlastnost se používá s vlastností **QMANAGER** . Tabulka CCDT obsahuje seznam kanálů klienta, které se používají pro připojení ke správci front definovanému v systému IBM MQ . Chcete-li získat informace o tom, jak jsou tabulky CCD používány produktem IBM MQ classes for JMS, prohlédněte si téma "Použití tabulky definic kanálů klienta s volbou IBM MQ classes for JMS" na stránce 275.

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 této vlastnosti 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 zakázáno.

QMGR
Produkt IBM MQ classes for JMS se pokusí připojit ke správci front se stejným identifikátorem správce front, ke kterému byl připojen, a to pomocí jedné z následujících voleb:

- Vlastnost **CONNECTIONNAMELIST** a kanál, který je definován ve vlastnosti **CHANNEL** .
- Tabulka CCDT definovaná ve vlastnosti **CCDTURL** .

ANY
Produkt IBM MQ classes for JMS se pokusí znovu připojit ke správci front se stejným názvem s použitím vlastnosti **CONNECTIONNAMELIST** nebo vlastnosti **CCDTURL**.

Související informace
Sekce CHANNELS konfiguračního souboru klienta

Použití automatického opětovného připojení klienta v prostředí Java EE
Adaptér prostředků IBM MQ , který lze implementovat do prostředí Java EE (Java Platform, Enterprise Edition), a poskytovatel systému zpráv produktu WebSphere Application Server IBM MQ používají produkt IBM MQ classes for JMS ke komunikaci se správci front IBM MQ . Adaptér prostředků IBM MQ a poskytovatel systému zpráv produktu WebSphere Application Server IBM MQ poskytují řadu mechanizmů, které umožňují, aby se specifikace aktivace, porty modulu listener WebSphere Application Server a aplikace spuštěné v kontejnerech klienta automaticky znovu připojily ke správci front. Objekty EJB (Enterprise JavaBeans) a webové aplikace musí implementovat vlastní logiku opětovného připojení.

Poznámka: Automatické opětovné připojení klienta se specifikacemi aktivace pomocí funkce poskytované produktem IBM MQ classes for JMS není podporováno (viz "Automatické opětovné připojení klienta JMS" na stránce 276). Adaptér prostředků IBM MQ poskytuje vlastní mechanizmus pro opětovné připojení specifikací aktivace, pokud se správce front, ke kterému se připojovala specifikace aktivace, stane nedostupným.

Mechanizmus, který adaptér prostředků poskytuje, je řízen:

- IBM MQ Vlastnost adaptéru prostředků **reconnectionRetryCount**.
- IBM MQ Vlastnost adaptéru prostředků **reconnectionRetryInterval**.

- Vlastnost specifikace aktivace **connectionNameList**.

Další informace o těchto vlastnostech viz “Konfigurace pro vlastnosti objektu ResourceAdapter” na stránce 437.

Použití automatického opětovného připojení klienta v rámci metody onMessage() aplikace objektu typu message-driven bean nebo jakékoli jiné aplikace spuštěné v prostředí Java Platform, Enterprise Edition není podporováno. Aplikace musí implementovat vlastní logiku opětovného připojení, pokud se správce front, ke kterému se připojovala, stane nedostupným. Další informace viz téma “Implementace logiky opětovného připojení v aplikaci Java EE” na stránce 290.

Podpora automatického opětovného 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 řadu mechanizmů, které aplikacím umožňují automatické opětovné připojení ke správci front. V některých případech se však na tuto podporu vztahují omezení.

Adaptér prostředků IBM MQ , který lze implementovat do prostředí Java EE a poskytovatele systému zpráv produktu WebSphere Application Server IBM MQ , používá ke komunikaci se správci front IBM MQ adaptér IBM MQ classes for JMS .

Následující tabulka shrnuje podporu, kterou adaptér prostředků IBM MQ a poskytovatel systému zpráv produktu WebSphere Application Server IBM MQ poskytují pro automatické opětovné připojení klienta.

Tabulka 45. Souhrn podpory pro volby automatického opětovného připojení klienta v prostředí Java EE

Volby pro automatické opětovné připojení	**vlastnost CONNECTIONNAMELIST**	**Vlastnost CCDTURL**	**Vlastnost CLIENTRECONNECTOPTIONS**	**Alternativní přístup k automatickému opětovnému připojení klienta**
Specifikace aktivace	Podporováno s omezeními	Podporováno s omezeními	Nepodporováno	Specifikace prostředí Java EE a aktivace poskytují vlastní mechanizmus opětovného připojení.
WebSphere Application Server portů modulu listener	Podporováno s omezeními	Podporováno s omezeními	Nepodporováno	Produkt WebSphere Application Server poskytuje vlastní mechanizmus opětovného připojení.
Podnikové JavaBeans a webové aplikace	Podporováno s omezeními	Podporováno s omezeními	Nepodporováno	Aplikace musí implementovat vlastní logiku opětovného připojení
Aplikace spuštěné v kontejnerech klienta	Podporováno	Podporováno	Podporováno	Nelze použít

Aplikace objektů typu message-driven bean, které jsou nainstalovány v prostředí Java EE , jako např. IBM MQ classes for JMS, mohou používat specifikace aktivace ke zpracování zpráv v systému IBM MQ . Specifikace aktivace se používají ke zjištění zpráv, které přicházejí do systému IBM MQ , a k jejich doručení do objektů typu message-driven bean ke zpracování. Objekty typu message-driven bean mohou také

vytvořit více připojení k systémům IBM MQ z jejich metody **onMessage()** . Další informace o tom, jak mohou tato připojení používat automatické opětovné připojení klienta, naleznete v tématu Enterprise JavaBeans a webové aplikace.

Specifikace aktivace
Pro specifikace aktivace jsou vlastnosti **CONNECTIONNAMELIST** a **CCDTURL** podporovány s omezeními a vlastnost **CLIENTRECONNECTOPTIONS** není podporována.

Aplikace objektů typu message-driven bean (MDB), které jsou nainstalovány v prostředí Java EE , jako např. WebSphere Application Server, mohou použít specifikace aktivace ke zpracování zpráv v systému IBM MQ .

Specifikace aktivace se používají ke zjištění zpráv přicházejících do systému IBM MQ a k jejich doručení do objektů MDB ke zpracování. Tento oddíl se zabývá tím, jak specifikace aktivace monitoruje systém IBM MQ .

Objekty MDB mohou také vytvořit další připojení k systémům IBM MQ zevnitř své metody `onMessage()` .

Podrobnosti o tom, jak mohou tato připojení používat automatické opětovné připojení klienta, naleznete v části “Podnikové JavaBeans a webové aplikace” na stránce 287.

CONNECTIONNAMELIST vlastnost

Při spuštění se specifikace aktivace pokusí připojit ke správci front pomocí následujícího příkazu:

- Jedna uvedená ve vlastnosti **QMANAGER**
- Kanál zmíněný ve vlastnosti **CHANNEL**
- Informace o názvu hostitele a portu z první položky v souboru **CONNECTIONNAMELIST**

Pokud se specifikace aktivace nemůže připojit ke správci front pomocí první položky v seznamu, přesune se specifikace aktivace na druhou položku atd., dokud nebude vytvořeno buď připojení ke správci front, nebo dokud nebude dosaženo konce seznamu.

Pokud se specifikace aktivace nemůže připojit k určenému správci front pomocí libovolné položky v souboru **CONNECTIONNAMELIST**, specifikace aktivace se zastaví a musí být restartována.

Po spuštění specifikace aktivace obdrží specifikace aktivace zprávy ze systému IBM MQ a doručí je do objektu MDB ke zpracování.

Pokud při zpracování zprávy dojde k selhání správce front, prostředí Java EE zjistí selhání a pokusí se znovu připojit specifikaci aktivace.

Specifikace aktivace používá informace ve vlastnosti **CONNECTIONNAMELIST** jako dříve, když specifikace aktivace provádí pokusy o opětovné připojení.

Pokud se specifikace aktivace pokusí o všechny položky v produktu **CONNECTIONNAMELIST** a stále se nemůže připojit ke správci front, čeká specifikace aktivace před dalším pokusem po dobu určenou vlastností IBM MQ adaptéru prostředků **reconnectionRetryInterval** .

IBM MQ Vlastnost adaptéru prostředků **reconnectionRetryCount** definuje počet následných pokusů o opětovné připojení, které se provedou před zastavením specifikace aktivace, a vyžaduje ruční restartování.

Po opětovném připojení specifikace aktivace k systému IBM MQ provede prostředí produktu Java EE požadované transakční vyčištění a pokračuje v doručování zpráv do objektů MDB za účelem zpracování.

Aby čištění transakcí fungovalo správně, prostředí Java EE musí mít přístup 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, musí soubor **CONNECTIONNAMELIST** obsahovat položku pro aktivní i rezervní instanci správce front.

To znamená, že prostředí Java EE může přistupovat k protokolům správce front, pokud prostředí potřebuje provést zotavení transakcí, bez ohledu na to, ke kterému správci front se prostředí znovu připojí po selhání.

Pokud jsou transakční objekty MDB používány se samostatnými správci front, musí vlastnost **CONNECTIONNAMELIST** obsahovat jedinou položku, aby se zajistilo, že se specifikace aktivace po selhání vždy znovu připojí ke stejnému správci front spuštěnému ve stejném systému.

CCDTURL vlastnost

Při spuštění se specifikace aktivace pokusí připojit ke správci front určenému ve vlastnosti **QMANAGER** pomocí první položky v tabulce CCDT (Client Channel Definition Table).

Pokud se specifikace aktivace nemůže připojit ke správci front pomocí první položky v tabulce, přesune se specifikace aktivace na druhou položku atd., dokud nebude vytvořeno připojení ke správci front nebo dokud nebude dosaženo konce tabulky.

Pokud se specifikace aktivace nemůže připojit k určenému správci front pomocí některé z položek v tabulce CCDT, specifikace aktivace se zastaví a musí být restartována.

Po spuštění specifikace aktivace obdrží specifikace aktivace zprávy ze systému IBM MQ a doručí je do objektu MDB ke zpracování.

Pokud při zpracování zprávy dojde k selhání správce front, prostředí Java EE zjistí selhání a pokusí se znovu připojit specifikaci aktivace.

Specifikace aktivace používá informace ve 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 před dalším pokusem čekat po dobu určenou vlastností IBM MQ adaptéru prostředků **reconnectionRetryInterval** .

IBM MQ Vlastnost adaptéru prostředků **reconnectionRetryCount** definuje počet následných pokusů o opětovné připojení, které se provedou před zastavením specifikace aktivace, a vyžaduje ruční restartování.

Po opětovném připojení specifikace aktivace k systému IBM MQ provede prostředí produktu Java EE požadované transakční vyčištění a pokračuje v doručování zpráv do objektů MDB za účelem zpracování.

Aby čištění transakcí fungovalo správně, prostředí Java EE musí mít přístup 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, musí tabulka CCDT obsahovat položku pro aktivní i rezervní instanci správce front.

To znamená, že prostředí Java EE může přistupovat k protokolům správce front, pokud prostředí potřebuje provést zotavení transakcí, bez ohledu na to, ke kterému správci front se prostředí znovu připojí po selhání.

Pokud jsou transakční objekty MDB používány se samostatnými správci front, musí tabulka CCDT obsahovat jedinou položku, aby se zajistilo, že specifikace aktivace se po selhání vždy znovu připojí ke stejnému správci front spuštěnému ve stejném systému.

Ujistěte se, že jste nastavili výchozí hodnotu *PREFERRED* pro vlastnost **AFFINITY** v tabulce CCDT, která se používá se specifikacemi aktivace, aby byla vytvořena připojení ke stejnému aktivnímu správci front.

CLIENTRECONNECTOPTIONS vlastnost

Specifikace aktivace poskytují vlastní funkčnost opětovného připojení. Poskytnutá funkce umožňuje specifikacím automatické opětovné připojení k systému IBM MQ v případě, že správce front, ke kterému byly připojeny, selže.

Z tohoto důvodu není funkce automatického opětovného připojení klienta poskytovaná produktem IBM MQ classes for JMS podporována.

Musíte nastavit vlastnost **CLIENTRECONNECTOPTIONS** na hodnotu *DISABLED* pro všechny specifikace aktivace, které se používají v produktu Java EE.

WebSphere Application Server portů modulu listener
Aplikace objektů typu message-driven bean (MDB) instalované v produktu WebSphere Application Server mohou také používat porty modulu listener ke zpracování zpráv v systému IBM MQ .

Porty modulu listener slouží ke zjištění zpráv přicházejících do systému IBM MQ a k jejich doručení do objektů MDB ke zpracování. Toto téma vysvětluje, jak port modulu listener monitoruje systém IBM MQ .

Objekty MDB mohou také vytvořit další připojení k systémům IBM MQ zevnitř své metody `onMessage()` .

Další informace o tom, jak tato připojení mohou používat automatické opětovné připojení klienta, naleznete v části "Podnikové JavaBeans a webové aplikace" na stránce 287 .

Pro porty modulu listener WebSphere Application Server :

- **CONNECTIONNAMELIST** a **CCDTURL** jsou podporovány s omezeními
- **CLIENTRECONNECTOPTIONS** není podporován

CONNECTIONNAMELIST vlastnost

Porty modulu listener používají fondy připojení JMS při připojování k produktu IBM MQ, takže podléhají důsledkům použití fondů připojení. Další informace viz "Specifikace aktivace" na stránce 284.

Pokud nejsou k dispozici žádná volná připojení a z této továrny připojení dosud nebyl vytvořen maximální počet připojení, vlastnost **CONNECTIONNAMELIST** se použije k pokusu o vytvoření nového připojení k produktu IBM MQ.

Pokud nejsou všechny systémy IBM MQ v produktu **CONNECTIONNAMELIST** přístupné, port modulu listener se zastaví.

Port modulu listener pak čeká po dobu určenou přizpůsobenou vlastností **RECOVERY.RETRY.INTERVAL** služby modulu listener pro zprávy a pokusí se znovu připojit.

Tento pokus o opětovné připojení zkontroluje, zda jsou ve fondu připojení nějaká volná připojení, pouze v případě, že byla vrácena mezi pokusy o připojení. Pokud 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 nezbytné transakční vyčištění a poté pokračuje v doručování zpráv do objektů MDB ke zpracování.

Aby čištění transakcí fungovalo správně, prostředí Java EE musí mít přístup 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**, musí soubor **CONNECTIONNAMELIST** obsahovat položku pro aktivní i rezervní instanci správce front.

To znamená, že prostředí Java EE může přistupovat k protokolům správce front, pokud prostředí potřebuje provést zotavení transakcí, bez ohledu na to, ke kterému správci front se prostředí znovu připojí po selhání.

Pokud jsou transakční objekty MDB používány se samostatnými správci front, musí vlastnost **CONNECTIONNAMELIST** obsahovat jedinou položku, aby se zajistilo, že se specifikace aktivace po selhání vždy znovu připojí ke stejnému správci front spuštěnému ve stejném systému.

CCDTURL vlastnost

Při spuštění se port modulu listener pokusí připojit ke správci front zadanému ve vlastnosti **QMANAGER** pomocí první položky v tabulce CCDT.

Pokud se port modulu listener nemůže připojit ke správci front pomocí první položky v tabulce, přesune se port modulu listener na druhou položku atd. dokud nebude vytvořeno připojení ke správci front nebo nebude dosaženo konce tabulky.

Pokud se port modulu listener nemůže připojit k určenému správci front pomocí žádné z položek v tabulce CCDT, port modulu listener se zastaví.

Port modulu listener pak čeká po dobu určenou přizpůsobenou vlastností **RECOVERY.RETRY.INTERVAL** služby modulu listener pro zprávy a pokusí se znovu připojit.

Tento pokus o opětovné připojení projde všemi položkami v tabulce CCDT jako dříve.

Jakmile je port modulu listener spuštěn, získá zprávy ze systému IBM MQ a doručí je do objektu MDB ke zpracování.

Pokud správce front selže při zpracování zprávy, prostředí Java EE zjistí selhání a pokusí se znovu připojit port modulu listener. Port modulu listener používá informace v tabulce CCDT při provádění pokusů o opětovné připojení.

Pokud se port modulu listener pokusí o všechny položky v tabulce CCDT a stále se nemůže připojit ke správci front, počká port před dalším pokusem po dobu určenou vlastností **RECOVERY.RETRY.INTERVAL** .

Vlastnost služby listener pro zprávy **MAX.RECOVERY.RETRIES** definuje počet po sobě jdoucích pokusů o opětovné připojení, které jsou provedeny před zastavením portu modulu listener a vyžadují ruční restart.

Jakmile se port modulu listener znovu připojí k systému IBM MQ , prostředí Java EE provede veškeré nezbytné transakční vyčištění a poté pokračuje v doručování zpráv do objektů MDB ke zpracování.

Aby čištění transakcí fungovalo správně, prostředí Java EE musí mít přístup 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, musí tabulka CCDT obsahovat položku pro aktivní i záložní instanci správce front.

To znamená, že prostředí Java EE může přistupovat k protokolům správce front, pokud prostředí potřebuje provést zotavení transakcí, bez ohledu na to, ke kterému správci front se prostředí znovu připojí po selhání.

Pokud jsou transakční objekty MDB používány se samostatnými správci front, musí tabulka CCDT obsahovat jedinou položku, aby bylo zajištěno, že port modulu listener bude po selhání vždy znovu připojen ke stejnému správci front spuštěnému ve stejném systému.

Ujistěte se, že jste nastavili výchozí hodnotu *PREFERRED* pro vlastnost **AFFINITY** na discích CCDT používaných s porty modulu listener, aby byla vytvořena připojení ke stejnému aktivnímu správci front.

CLIENTRECONNECTOPTIONS vlastnost

Porty modulu listener poskytují vlastní funkčnost opětovného připojení. Poskytnutá funkčnost umožňuje portům modulu listener automatické opětovné připojení k systému IBM MQ v případě, že dojde k selhání správce front, ke kterému byly připojeny.

Z tohoto důvodu není funkce automatického opětovného připojení klienta poskytovaná produktem 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 JavaBeans a webové aplikace
Aplikace a aplikace EJB (Enterprise JavaBean), které jsou spuštěny v rámci webového kontejneru, například Servlety, používají továrnu připojení JMS k vytvoření připojení ke správci front IBM MQ .

Pro objekty EJB a webové aplikace platí následující omezení:

- **CONNECTIONNAMELIST** a **CCDTURL** jsou podporovány s omezeními

- **CLIENTRECONNECTOPTIONS** není podporován

CONNECTIONNAMELIST vlastnost

Pokud prostředí Java EE poskytuje fond připojení pro připojení JMS, informace o tom, jak to ovlivňuje chování vlastnosti **CONNECTIONNAMELIST** , naleznete v části “Použití CONNECTIONNAMELIST nebo CCDT ve fondu připojení” na stránce 289 .

Pokud prostředí Java EE neposkytuje fond připojení JMS . aplikace používá vlastnost **CONNECTIONNAMELIST** stejným způsobem jako aplikace Java SE .

Pokud jsou aplikace používány s transakčními objekty MDB, které se účastní transakcí XA a připojují se ke správci front s více instancemi, musí soubor **CONNECTIONNAMELIST** obsahovat položku pro aktivní i rezervní instanci správce front.

To znamená, že prostředí Java EE může přistupovat k protokolům správce front, pokud prostředí potřebuje provést zotavení transakcí, bez ohledu na to, ke kterému správci front se prostředí znovu připojí po selhání.

Pokud jsou aplikace používány se samostatnými správci front, musí vlastnost **CONNECTIONNAMELIST** obsahovat jedinou položku, aby bylo zajištěno, že se aplikace po selhání vždy znovu připojí ke stejnému správci front spuštěnému ve stejném systému.

CCDTURL vlastnost

Pokud prostředí Java EE poskytuje fond připojení pro připojení JMS , informace o tom, jak to ovlivňuje chování vlastnosti **CCDTURL** , naleznete v části “Použití CONNECTIONNAMELIST nebo CCDT ve fondu připojení” na stránce 289 .

Pokud prostředí Java EE neposkytuje fond připojení JMS . aplikace používá vlastnost **CCDTURL** stejným způsobem jako aplikace Java SE .

Pokud jsou aplikace používány s transakčními objekty MDB, které se účastní transakcí XA a připojují se ke správci front s více instancemi, musí tabulka CCDT obsahovat položku pro aktivní i rezervní instanci správce front.

To znamená, že prostředí Java EE může přistupovat k protokolům správce front, pokud prostředí potřebuje provést zotavení transakcí, bez ohledu na to, ke kterému správci front se prostředí znovu připojí po selhání.

Pokud jsou aplikace používány se samostatnými správci front, musí tabulka CCDT obsahovat jedinou položku, aby bylo zajištěno, že se specifikace aktivace po selhání vždy znovu připojí ke stejnému správci front spuštěnému ve stejném systému.

CLIENTRECONNECTOPTIONS vlastnost

Vlastnost **CLIENTRECONNECTOPTIONS** musíte nastavit na hodnotu *DISABLED* pro všechny továrny připojení JMS používané objekty EJB nebo aplikacemi, které jsou spuštěny ve webovém kontejneru.

Aplikace, které vyžadují automatické opětovné připojení k novému správci front, musí v případě, že správce front, který používají, selže, implementovat vlastní logiku opětovného připojení. Další informace viz “Implementace logiky opětovného připojení v aplikaci Java EE” na stránce 290.

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

Aplikace spuštěné v kontejnerech klienta
Některá prostředí Java EE , jako např. WebSphere Application Server, poskytují kontejner klienta, který lze použít ke spouštění aplikací Java SE .

Aplikace spuštěné v těchto prostředích používají továrnu připojení JMS pro připojení ke správci front IBM MQ .

Pro aplikace spuštěné v kontejnerech klienta:

- **CONNECTIONNAMELIST** a **CCDTURL** jsou plně podporovány
- **CLIENTRECONNECTOPTIONS** je plně podporován

CONNECTIONNAMELIST vlastnost

Pokud prostředí Java EE poskytuje fond připojení pro připojení JMS, informace o tom, jak to ovlivňuje chování vlastnosti **CONNECTIONNAMELIST** , naleznete v části “Použití CONNECTIONNAMELIST nebo CCDT ve fondu připojení” na stránce 289 .

Pokud prostředí Java EE neposkytuje fond připojení JMS . aplikace používá vlastnost **CONNECTIONNAMELIST** stejným způsobem jako aplikace Java SE .

CCDTURL vlastnost

Pokud prostředí Java EE poskytuje fond připojení pro připojení JMS , informace o tom, jak to ovlivňuje chování vlastnosti **CCDTURL** , naleznete v části “Použití CONNECTIONNAMELIST nebo CCDT ve fondu připojení” na stránce 289 .

Pokud prostředí Java EE neposkytuje fond připojení JMS . aplikace používá vlastnost **CCDTURL** stejným způsobem jako aplikace Java SE .

Použití CONNECTIONNAMELIST nebo CCDT ve fondu připojení
Některá prostředí Java EE , například WebSphere Application Server, poskytují fond připojení JMS . kontejner, který lze použít ke spuštění aplikací Java SE .

Aplikace, které vytvářejí připojení pomocí továrny připojení, která byla definována v prostředí Java EE , buď získají existující volné připojení z fondu připojení pro tuto továrnu připojení, nebo nové připojení, pokud ve fondu připojení není vhodné.

To může mít důsledky, pokud byla továrna připojení konfigurována s definovanou vlastností **CONNECTIONNAMELIST** nebo **CCDTURL** .

Při prvním použití továrny připojení k vytvoření připojení prostředí Java EE používá buď produkt **CONNECTIONNAMELIST**. nebo **CCDTURL** pro vytvoření nového připojení k systému IBM MQ . Pokud toto připojení již není vyžadováno, bude vráceno do fondu připojení, kde bude k dispozici pro opětovné použití.

Pokud něco jiného vytvoří připojení z továrny připojení, prostředí Java EE vrátí připojení z fondu připojení namísto použití vlastností **CONNECTIONNAMELIST** nebo **CCDTURL** k vytvoření nového připojení.

Pokud je připojení používáno v případě selhání instance správce front, bude připojení zrušeno. Obsah fondu připojení však nemusí být, což znamená, že fond může potenciálně i nadále obsahovat připojení ke správci front, který již není spuštěn.

V této situaci je při příštím požadavku na vytvoření připojení z továrny připojení vráceno připojení k nezdařenému správci front. Veškeré pokusy o použití tohoto připojení se nezdaří, protože správce front již není spuštěn, což způsobí zrušení připojení.

Pouze v případě, že je fond připojení prázdný, použije prostředí Java EE k vytvoření nového připojení k produktu IBM MQvlastnosti **CONNECTIONNAMELIST** nebo **CCDTURL** .

Vzhledem k tomu, jak se **CONNECTIONNAMELIST** a tabulky CCDT používají k vytváření připojení JMS , je také možné mít fond připojení, který obsahuje připojení k různým systémům IBM MQ .

Předpokládejme například, že továrna připojení byla nakonfigurována 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 správce front spuštěný v systému `hostname1(port1)` přístupný. To znamená, že aplikace skončí s připojením ke správci front spuštěnému v systému `hostname2(port2)`.

Nyní přichází další aplikace a vytvoří připojení JMS ze stejné továrny připojení. Správce front v systému `hostname1(port1)` je nyní k dispozici, takže se vytvoří nové připojení JMS k tomuto systému IBM MQ a vrátí se do aplikace.

Po dokončení obou aplikací dojde k zavření produktu JMS Connections, což způsobí vrácení připojení do fondu připojení.

Výsledkem je, že fond 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 transakčním protokolům.

Implementace logiky opětovného připojení v aplikaci Java EE
Podnikové JavaBeans a webové aplikace, které se chtějí automaticky znovu připojit, pokud se správci front nepodaří implementovat vlastní logiku opětovného připojení.

Následující volby poskytují další informace o tom, jak toho dosáhnout.

Povolit selhání aplikace

Tento přístup nevyžaduje žádné změny aplikace, ale vyžaduje administrativní opětovnou konfiguraci definice továrny připojení, aby obsahovala vlastnost **CONNECTIONNAMELIST** . Tento přístup však vyžaduje, aby byl původce volání schopen správně zpracovat selhání. Všimněte si, že toto je také nezbytné pro selhání, jako např. MQRC_Q_FULL, která nesouvisí se selháním připojení.

Vzorový 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 pro továrnu připojení, kterou tento servlet používá, je definována vlastnost **CONNECTIONNAMELIST** .

Při prvním zpracování servletu je vytvořeno nové připojení pomocí vlastnosti **CONNECTIONNAMELIST** za předpokladu, že z jiných aplikací, které se připojují ke stejnému správci front, nejsou k dispozici žádná připojení ve fondu.

Po uvolnění připojení 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. V tomto okamžiku je vygenerována událost CONNECTION_ERROR_OCCURS. Tato událost vyzve fond ke zrušení nezdařených připojení.

Při příštím spuštění aplikace není 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 záložní instanci, jakmile bude k dispozici.

Pokud jsou do aplikace zapojeny jiné prostředky, například databáze, může být vhodné označit, že by aplikační server měl transakci odvolat.

Zpracovat opětovné připojení v rámci aplikace

Pokud původce volání nemůže zpracovat selhání ze servletu, musí být opětovné připojení zpracováno v rámci aplikace. Jak je ukázáno v následujícím příkladu, zpracování opětovného připojení v rámci aplikace vyžaduje, aby aplikace požadovala nové připojení, aby mohla uložit do mezipaměti továrnu připojení, kterou vyhledala v produktu JNDI , a zpracovat volání `JMSException` , jako např. `JMSCMQ0001: WebSphere MQ se nezdařilo volání compcode '2' ('MQCC_FAILED '), příčina' 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 zahrne smyčku spánku a opakování pro zpracování neokamžitého opětovného připojení. V praxi tato metoda zabraňuje mnoha pokusům o opětovné připojení. Všimněte si, že podmínky ukončení byly z důvodu přehlednosti vynechány.

```
 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 uvolnila všechna připojení, která jsou zadržena pro jiné prostředky, ať už se jedná o jiné správce front IBM MQ nebo jiné služby back-end, jako jsou databáze. Po dokončení opětovného připojení k nové instanci správce front IBM MQ je třeba tato připojení znovu vytvořit. Pokud připojení znovu neustanovíte, budou prostředky aplikačního serveru

během pokusu o opětovné připojení zbytečně zadrženy a může dojít k vypršení časového limitu v době, kdy jsou znovu použity.

Použití správce WorkManager

Pro aplikace s dlouhou životností (například dávkové zpracování), jejichž doba zpracování je delší než několik desítek sekund, lze použít 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 je nyní implementována prostřednictvím pracovního 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 dávky trvá dlouhou dobu, například velké zprávy, pomalá síť nebo rozsáhlý přístup k databázi (zejména v kombinaci s pomalým překonáním selhání), začne server vypisovat varování zablokovaného podprocesu, podobně jako v následujícím příkladu:

WSVR0605W: Podproces "`WorkManager.DefaultWorkManager` : 0" (00000035) byl aktivní po dobu 694061 milisekund a může být zablokovaný. Na serveru je celkem 1 podproces (y), které mohou být zablokované.

Tato varování lze minimalizovat snížením velikosti dávky nebo zvýšením časového limitu zablokovaného podprocesu. Obecně je však vhodnější, pokud implementujete toto zpracování do zpracování EJB (pro dávkové odeslání) nebo objektu typu message-driven bean (pro příjem nebo příjem a odpověď).

Všimněte si, že opětovné připojení spravované aplikací neposkytuje obecné řešení pro zpracování běhových chyb a aplikace musí stále zpracovávat chyby, které nesouvisejí se selháním připojení.

Například pokus o vložení zprávy do fronty, která je plná (2053 MQRC_Q_FULL), nebo pokus o připojení ke správci front pomocí neplatných pověření zabezpečení (2035 MQRC_NOT_AUTHORIZED).

Aplikace musí také zpracovat chyby 2059 MQRC_Q_MGR_NOT_AVAILABLE, když nejsou okamžitě k dispozici žádné instance, když probíhá překonání selhání. Toho lze dosáhnout aplikací, která nahlásí výjimky JMS tak, jak se vyskytnou, namísto bezobslužného pokusu o opětovné připojení.

IBM MQ classes for JMS sdružování objektů
Použití formy sdružování připojení mimo produkt Java EE pomáhá snížit celkovou zátěž vyplývající například z některých samostatných aplikací, které používají rámce, nebo z implementace do cloudových prostředí, a také z většího počtu klientských připojení do produktu `QueueManagers` , což vede ke zvýšení konsolidace aplikací a správců front na serveru.

V programovacím modelu Java EE existuje dobře definovaný životní cyklus různých používaných objektů. Objekty typu message-driven bean (MDB) jsou nejvíce omezeny, zatímco servlety poskytují větší svobodu. Proto volby sdružování, které jsou k dispozici v rámci serverů Java EE , odpovídají různým používaným programovacím modelům.

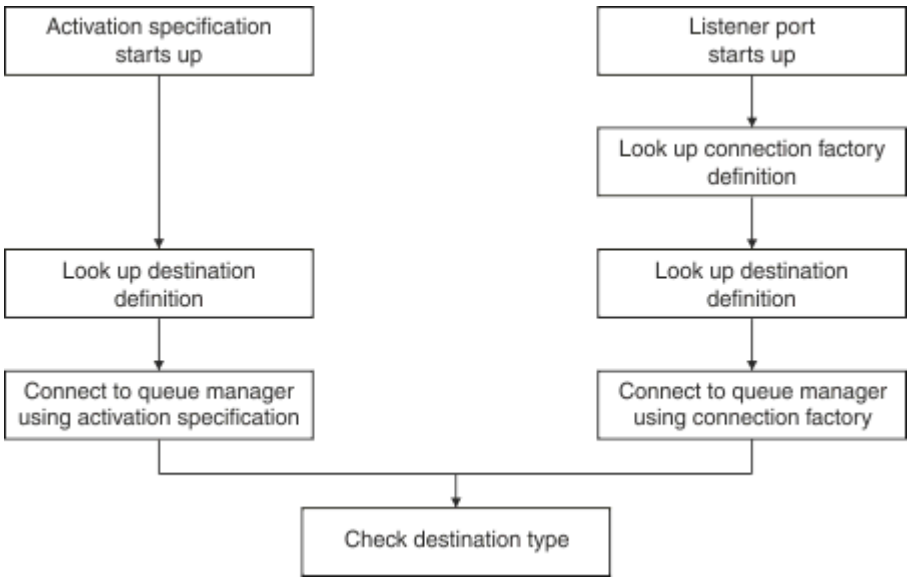
S produktem Java SE (nebo s jiným rámcem, jako je například Spring) jsou programovací modely extrémně flexibilní. Proto jednotná strategie sdružování nevyhovuje všem. Měli byste zvážit, zda existuje rámec, který by mohl provádět jakoukoli formu sdružování, například jaro.

Strategie sdružování, která se má použít, závisí na prostředí, ve kterém je aplikace spuštěna.

Sdružování objektů v prostředí Java EE
Aplikační servery Java EE poskytují funkce sdružování připojení, které mohou používat aplikace objektů typu message-driven bean, objekty enterprise Java bean a servlety.

Produkt WebSphere Application Server udržuje fond připojení k poskytovateli JMS , aby se zlepšil výkon. Když aplikace vytvoří připojení JMS , aplikační server určí, zda připojení již existuje ve fondu volných připojení. Pokud ano, připojení se vrátí do aplikace; jinak se vytvoří nové připojení.

Obrázek 41 na stránce 294 ukazuje, jak specifikace aktivace a porty modulu listener vytvářejí připojení JMS a používají toto připojení k monitorování místa určení pro zprávy v normálním režimu.



Obrázek 41. Normální režim

Při použití poskytovatele systému zpráv IBM MQ mohou aplikace, které provádějí odchozí systém zpráv (například objekty enterprise Java Bean a servlety) a komponenta portu modulu listener objektu typu message-driven bean, používat tyto fondy připojení.

Specifikace aktivace poskytovatele systému zpráv IBM MQ používají funkci sdružování připojení poskytovanou adaptérem prostředků IBM MQ . Další informace naleznete v tématu Konfigurace vlastností pro adaptér prostředků WebSphere MQ .

“Příklady použití fondu připojení” na stránce 297 vysvětluje, jak aplikace, které provádějí odchozí systém zpráv, a porty modulu listener používají při vytváření připojení produktu JMS volný fond.

“Volné podprocesy údržby fondu připojení” na stránce 300 vysvětluje, co se stane s těmito připojeními, když aplikace nebo port modulu listener dokončí připojení.

“Příklady podprocesů údržby fondu” na stránce 302 vysvětluje, jak se vyčistí volný fond připojení, aby se zabránilo JMS zastarání připojení.

WebSphere Application Server má limit počtu připojení, která lze vytvořit z továrny, určený vlastností *maximum připojení* továrny připojení. Výchozí hodnota této vlastnosti je 10, což znamená, že z továrny může být vytvořeno až 10 připojení najednou.

Ke každé továrně je přidružen fond volných připojení. Při spuštění aplikačního serveru jsou volné fondy připojení prázdné. Vlastnost Maximální počet připojení, která mohou existovat ve fondu volných připojení pro továrnu, je také určena vlastností Maximální počet připojení.

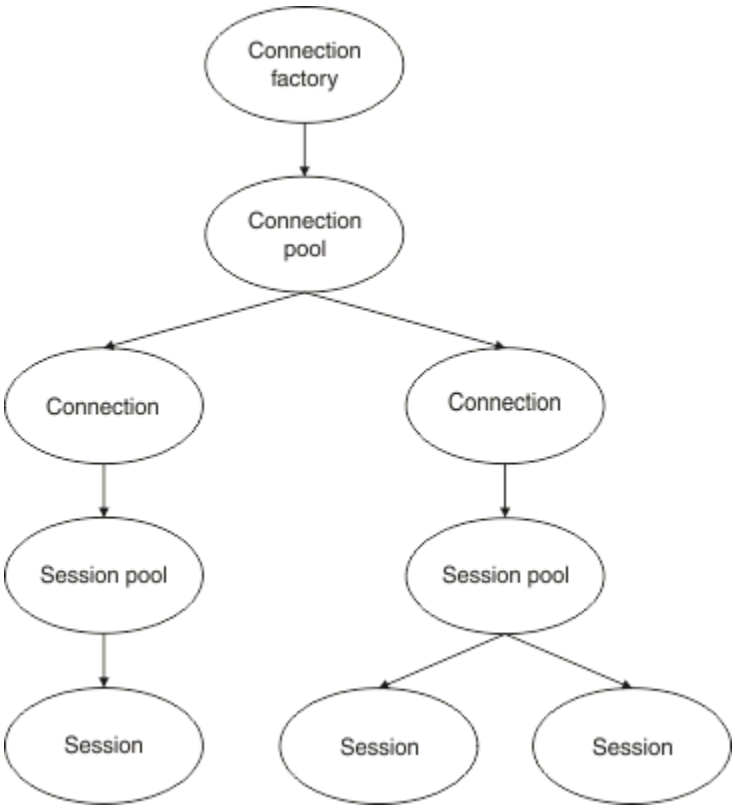
Tip: Pomocí produktu JMS 2.0lze továrnu připojení použít k vytvoření připojení i kontextů. V důsledku toho je možné, aby byl fond připojení přidružen k továrně připojení, která obsahuje kombinaci připojení i kontextů. Doporučuje se, aby se továrna připojení používala pouze pro vytváření připojení nebo kontextů. Tím se zajistí, že fond připojení pro danou továrnu připojení bude obsahovat pouze objekty jednoho typu, což tento fond zefektivní.

Informace o fungování sdružování připojení v produktu WebSphere Application Servernaleznete v tématu Konfigurace sdružování připojení pro připojení JMS. Další informace o aplikačních serverech naleznete v příslušné dokumentaci k aplikačnímu serveru.

Jak se používá fond připojení

Ke každé továrně 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 nula nebo více relací JMS .

Obrázek 42 na stránce 295 ukazuje vztah mezi těmito objekty.



Obrázek 42. Fondy připojení a fondy relací

Když se spustí port modulu listener nebo aplikace, která chce provést odchozí zasílání zpráv, použije továrnu k vytvoření připojení, port nebo aplikace zavolá 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 volného fondu pro tuto továrnu a vrátit jej do aplikace.

Pokud ve fondu nejsou žádná volná připojení a počet připojení vytvořených z této továrny nedosáhl limitu určeného ve vlastnosti *maximum connections* dané továrny, správce Connection Manager vytvoří nové připojení, které má aplikace používat.

Pokud se však aplikace pokusí vytvořit připojení, ale počet připojení vytvořených z této továrny se již rovná vlastnosti *maximum připojení* továrny, aplikace čeká na zpřístupnění připojení (které má být vloženo zpět do volného fondu).

Doba, po kterou aplikace čeká, je určena ve vlastnosti *timeout připojení* fondu připojení, která má výchozí hodnotu 180 sekund. Pokud je připojení vloženo zpět do volného fondu během tohoto období 180 sekund, správce Connection Manager jej okamžitě vyjme z fondu a předá jej aplikaci. Pokud však časový limit uplyne, dojde k výjimce *ConnectionWaitTimeoutException* .

Když aplikace dokončí připojení a zavře jej voláním:

- **Connection.close()**
- **QueueConnection.close()**
- **TopicConnection.close()**

připojení je ve skutečnosti ponecháno otevřené a je vráceno do volného fondu, aby jej mohla znovu použít jiná aplikace. Proto můžete mít otevřená připojení mezi produktem WebSphere Application Server a poskytovatelem JMS , a to i v případě, že na aplikačním serveru nejsou spuštěny žádné aplikace JMS .

Rozšířené vlastnosti fondu připojení
Existuje řada rozšířených vlastností, které lze použít k řízení chování fondů připojení JMS .

Přepěťová ochrana

"Způsob, jakým aplikace provádějící odchozí zasílání zpráv používají fond připojení" na stránce 299 popisuje použití metody `sendMessage()`, která zahrnuje `connectionFactory.createConnection()`.

Zvažte situaci, kdy máte 50 objektů EJB, které všechny vytvářejí připojení JMS ze stejné továrny připojení jako součást jejich metody `ejbCreate()` .

Pokud jsou všechny tyto objekty bean vytvořeny současně a ve fondu volných připojení továrny nejsou žádná připojení, pokusí se aplikační server vytvořit současně 50 připojení JMS ke stejnému poskytovateli JMS . Výsledkem je významné zatížení jak poskytovatele WebSphere Application Server , tak poskytovatele JMS .

Vlastnosti nárazové ochrany mohou této situaci zabránit omezením počtu připojení produktu JMS , která lze kdykoli vytvořit z továrny připojení, a rozložením vytváření dalších připojení.

Omezení počtu připojení JMS v libovolném okamžiku je dosaženo pomocí dvou vlastností:

- Prahová hodnota nárazového nárůstu
- Interval vytváření nárazového vytváření.

Když se aplikace EJB pokusí vytvořit připojení JMS z továrny připojení, správce připojení zkontroluje, kolik připojení se vytváří. Pokud je toto číslo menší nebo rovno hodnotě vlastnosti `surge threshold` , správce připojení pokračuje v otevírání nových připojení.

Pokud však počet vytvářených připojení přesahuje vlastnost `surge threshold` , správce připojení před vytvořením a otevřením nového připojení počká po dobu určenou vlastností `surge creation interval` .

Zablokované připojení

JMS Připojení je považováno za `stuck`, pokud aplikace JMS používá toto připojení k odeslání požadavku poskytovateli JMS a poskytovatel neodpoví během určité doby.

WebSphere Application Server poskytuje způsob, jak zjistit připojení produktu `stuck` JMS . Chcete-li použít tuto funkci, musíte nastavit tři vlastnosti:

- Časovač času zablokování
- Čas zablokování
- Prahová hodnota zablokování

"Příklady podprocesů údržby fondu" na stránce 302 vysvětluje, jak se podproces údržby fondu spouští pravidelně a kontroluje obsah volného fondu továrny připojení, hledá připojení, která buď byla po určitou dobu nevyužita, nebo existují příliš dlouho.

Chcete-li zjistit zablokované připojení, aplikační server také spravuje podproces zablokované připojení, který kontroluje stav všech aktivních připojení vytvořených z továrny připojení, aby zjistil, zda některá z nich čeká na odpověď od poskytovatele JMS .

Když se podproces připojení zablokování spustí, je určen vlastností `Stuck time timer` . Výchozí hodnota této vlastnosti je nula, což znamená, že detekce zablokované připojení se nikdy nespustí.

Pokud podproces najde odpověď čekající na odpověď, určí, jak dlouho čeká, a porovná tuto dobu s hodnotou vlastnosti `Stuck time` .

Pokud doba potřebná pro odpověď poskytovatele JMS překročí dobu určenou vlastností `Stuck time` , aplikační server označí připojení JMS jako zablokované.

Předpokládejme například, že továrna připojení `jms/CF1` má vlastnost `Stuck time timer` nastavenou na hodnotu 10 a vlastnost `Stuck time` nastavenou na hodnotu 15.

Zablokované vlákno připojení se stane aktivní každých 10 sekund a zkontroluje, zda jakékoli připojení vytvořené z produktu `jms/CF1` čekalo na odezvu z produktu IBM MQdéle než 15 sekund.

Předpokládejme, že EJB vytvoří JMS připojení k IBM MQ pomocí `jms/CF1`a poté se pokusí vytvořit JMS relaci používající toto připojení voláním funkce `Connection.createSession()`.

Něco však brání poskytovateli JMS v odpovědi na požadavek. Je možné, že došlo k zamrznutí počítače nebo že proces spuštěný na poskytovateli JMS je zablokován, což brání zpracování jakékoli nové práce:

Deset sekund po volání objektu typu EJB `Connection.createSession()`se časovač zablokované připojení stane aktivním a zobrazí se aktivní připojení vytvořená z produktu `jms/CF1`.

Předpokládejme, že existuje pouze jedno aktivní připojení, například s názvem `c1`. První objekt EJB čekal 10 sekund na odpověď na požadavek, který odeslal do produktu `c1`, což je méně než hodnota parametru `Stuck time`, takže časovač připojení zablokování toto připojení ignoruje a stane se neaktivním.

O 10 sekund později se podproces připojení zablokování znovu stane aktivním a vyhledá aktivní připojení pro produkt `jms/CF1`. Jako dříve předpokládejme, že existuje pouze jedno připojení, `c1`.

Od prvního objektu typu EJB s názvem `createSession()`je to nyní 20 sekund a objekt typu EJB stále čeká na odpověď. 20 sekund je delší než čas uvedený ve vlastnosti `Stuck time` , takže podproces připojení zablokování označí `c1` jako zablokovaný.

Pokud po pěti sekundách produkt IBM MQ nakonec odpoví a umožní prvnímu objektu EJB vytvořit relaci JMS , bude připojení opět používáno.

Aplikační server spočítá počet připojení JMS vytvořených z továrny připojení, která jsou zablokované. Když aplikace používá tuto továrnu připojení k vytvoření nového připojení JMS a ve volném fondu této továrny nejsou žádná volná připojení, správce připojení porovná počet zablokované připojení s hodnotou vlastnosti `Stuck threshold` .

Pokud je počet zablokované připojení menší než hodnota nastavená pro vlastnost `Stuck threshold`, vytvoří správce připojení nové připojení a předá jej aplikaci.

Pokud se však počet zablokované připojení rovná hodnotě vlastnosti `Stuck threshold` , aplikace obdrží výjimku prostředku.

Oblasti fondu

Produkt WebSphere Application Server poskytuje dvě vlastnosti, které vám umožňují rozdělit fond volných připojení pro továrnu připojení:

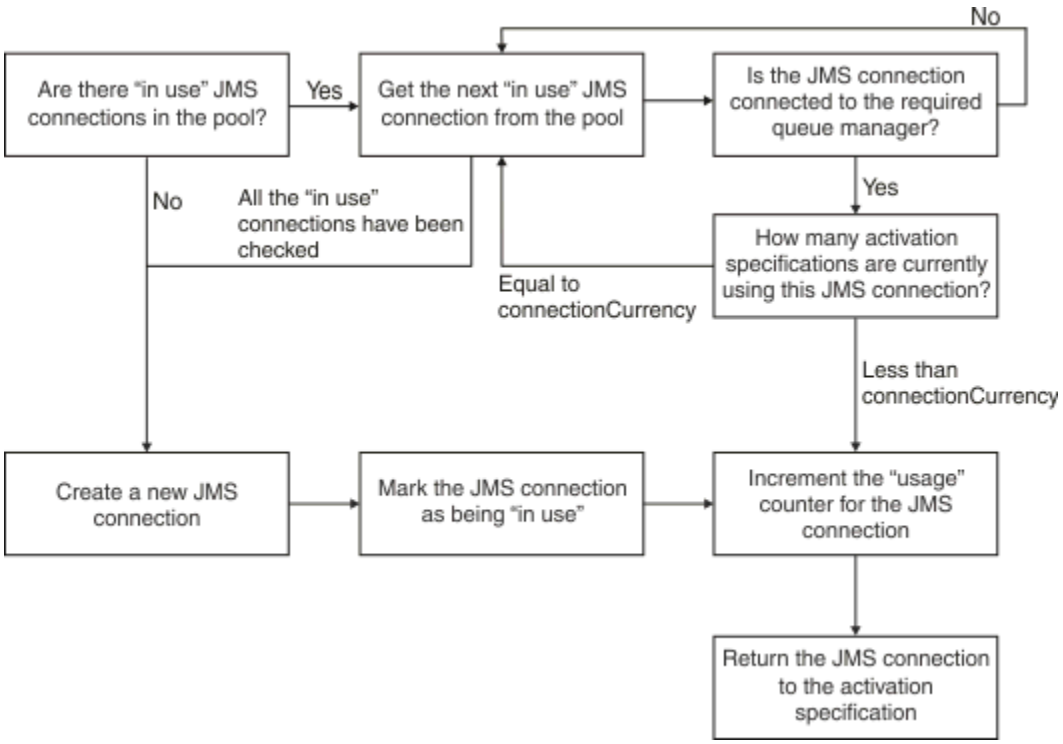
- `Number of free pool partitions` sděluje aplikačnímu serveru, do kolika oblastí chcete rozdělit fond volných připojení.
- `Free pool distribution table size` určuje, jak jsou oblasti indexovány.

Tyto vlastnosti ponechte na výchozích hodnotách nula, pokud vás centrum podpory IBM nepožádá o jejich změnu.

Všimněte si, že produkt WebSphere Application Server má jednu další rozšířenou vlastnost fondu připojení s názvem `Number of shared partitions`. Tato vlastnost určuje počet oblastí používaných k ukládání sdílených připojení. Vzhledem k tomu, že připojení JMS jsou vždy nesdílená, tato vlastnost se nepoužije.

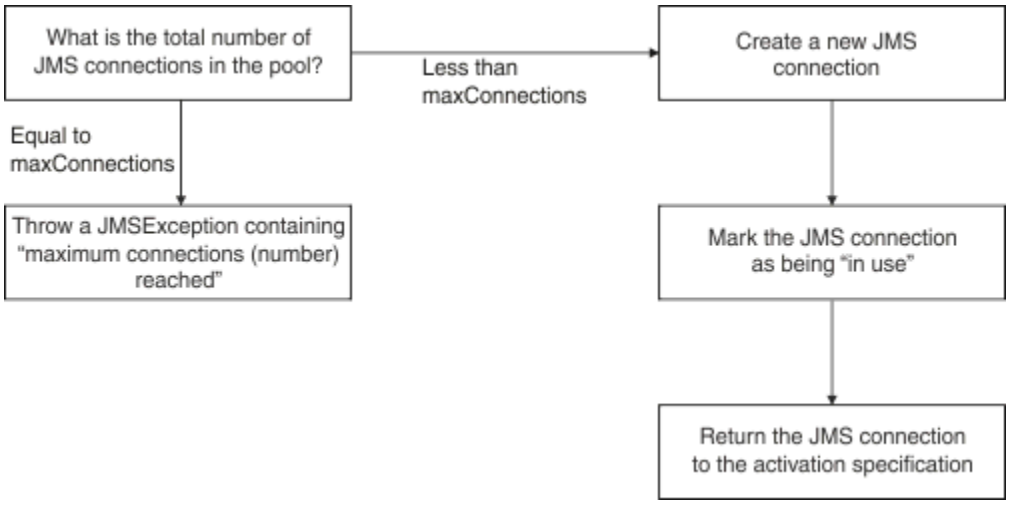
Příklady použití fondu připojení
Komponenta portu modulu listener objektu typu message-driven bean a aplikace, které provádějí odchozí systém zpráv, používají fond připojení JMS .

Obrázek 43 na stránce 298 zobrazuje, jak fond připojení funguje pro WebSphere Application Server V7.5 a V8.0.



Obrázek 43. WebSphere Application Server V7.5 a V8.0 -jak funguje fond připojení

Obrázek 44 na stránce 298 ukazuje, jak fond připojení funguje pro produkt WebSphere Application Server V8.5.



Obrázek 44. WebSphere Application Server V8.5 -Jak funguje fond připojení

Jak porty modulu listener MDB používají fond připojení

Předpokládejme, že máte implementovaný objekt MDB v systému WebSphere Application Server Network Deployment, který používá IBM MQ jako poskytovatele JMS . Objekt MDB je implementován na portu modulu listener, který používá továrnu připojení s názvem například `jms/CF1`, která má vlastnost *maximum connections* nastavenou na hodnotu 2, což znamená, že z této továrny lze v daném okamžiku vytvořit pouze dvě připojení.

Při spuštění portu modulu listener se port pokusí vytvořit připojení k produktu IBM MQpomocí továrny připojení produktu`jms/CF1` .

K tomu si port vyžádá připojení od správce připojení. Vzhledem k tomu, že se jedná o první použití továrny připojení `jms/CF1` , nejsou ve fondu volných připojení `jms/CF1` žádná připojení, takže správce připojení vytvoří nový, například s názvem `c1`. Všimněte si, že toto připojení existuje po celou dobu životnosti portu modulu listener.

Nyní zvažte situaci, kdy zastavíte port modulu listener pomocí administrativní konzoly WebSphere Application Server . V tomto případě správce připojení převezme připojení a vloží je 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 připojení (`c1`) ve fondu volných prostředků, správce připojení toto připojení z fondu vyjme a zpřístupní pro port modulu listener.

Nyní předpokládejme, že máte na aplikačním serveru implementován druhý objekt typu message-driven bean, který používá jiný port modulu listener.

Předpokládejme, že se pak pokusíte spustit třetí port modulu listener, který je také nakonfigurován pro použití továrny připojení `jms/CF1` . Třetí port modulu listener požaduje připojení od správce připojení, který hledá ve volném fondu pro produkt `jms/CF1` a zjistí, že je prázdný. Poté 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 že jste již vytvořili dvě připojení z této továrny, správce připojení počká 180 sekund (výchozí hodnota vlastnosti časového limitu připojení), než bude připojení k dispozici.

Pokud však zastavíte první port modulu listener, jeho připojení `c1` se vloží do volného fondu pro `jms/CF1`. Správce připojení načte toto připojení a předá je třetímu modulu listener.

Pokud se nyní pokusíte restartovat první modul listener, musí tento modul listener počkat na zastavení jednoho z ostatních portů modulu listener, než bude možné restartovat první modul listener. Pokud není žádný ze spuštěných portů modulu listener zastaven během 180 sekund, první modul listener obdrží chybu `ConnectionWaitTimeoutException` a zastaví se.

Způsob, jakým aplikace provádějící odchozí zasílání zpráv používají fond připojení

Pro tuto volbu předpokládejme, že na aplikačním serveru je nainstalován jeden objekt EJB s názvem, například `EJB1`. Objekt typu bean implementuje metodu s názvem `sendMessage()` pomocí:

- Vytvoření JMS připojení k IBM MQ z továrny `jms/CF1`pomocí `connectionFactory.createConnection()`.
- Vytvoření relace JMS z připojení.
- Vytvoření producenta zpráv z relace.
- Odeslání zprávy.
- Zavírání producenta.
- Probíhá zavírání relace.
- Zavírání připojení voláním funkce `connection.close()`.

Předpokládejme, že volný fond pro továrnu `jms/CF1` je prázdný. Při prvním vyvolání objektu typu EJB se objekt typu bean pokusí vytvořit připojení k objektu IBM MQ z továrny `jms/CF1`. Vzhledem k tomu, že volný fond pro továrnu je prázdný, správce připojení vytvoří nové připojení a předá jej produktu `EJB1`.

Těsně před ukončením metody volá metoda `connection.close()`. Namísto zavření `c1`správce připojení převezme připojení a vloží je do volného fondu pro produkt `jms/CF1`.

Při příštím volání metody `sendMessage()` vrátí metoda `connectionFactory.createConnection()` aplikaci hodnotu `c1` .

Předpokládejme, že máte druhou instanci EJB spuštěnou současně s první instancí. Když obě instance volají `sendMessage()`, vytvoří se dvě připojení z továrny připojení `jms/CF1` .

Nyní předpokládejme, že je vytvořena třetí instance objektu bean. Když třetí objekt typu bean vyvolá volání `sendMessage()`, metoda vyvolá volání `connectionFactory.createConnection()` pro vytvoření připojení z `jms/CF1`.

V současné době však existují dvě připojení vytvořená z produktu `jms/CF1`, která se rovnají hodnotě maximálního počtu připojení pro tuto továrnu. Proto metoda `createConnection()` čeká na zpřístupnění připojení po dobu 180 sekund (výchozí hodnota vlastnosti časového limitu připojení).

Pokud však metoda `sendMessage()` pro první volání EJB `connection.close()` a skončí, připojení, které používala, `c1`, se vloží zpět do volného fondu připojení. Správce připojení převezme připojení zpět z volného fondu a předá jej třetímu objektu EJB. Volání z tohoto objektu typu bean do objektu `connectionFactory.createConnection()` se poté vrátí a umožní dokončení metody `sendMessage()`.

Porty modulu listener MDB a objekty EJB používající stejný fond připojení

Dva předchozí příklady ukazují, jak mohou porty modulu listener a objekty EJB používat fond připojení izolovaně. Můžete však mít port modulu listener i objekt EJB spuštěný ve stejném aplikačním serveru a vytvářet připojení JMS pomocí stejné továrny připojení.

Musíte zvážit důsledky této situace.

Je třeba si uvědomit, že továrna připojení je sdílena mezi portem modulu listener a objektem EJB.

Předpokládejme například, že máte modul listener a objekt EJB spuštěné současně. Obě používají továrnu připojení `jms/CF1` , což znamená, že bylo dosaženo limitu připojení určeného vlastností maximálního počtu připojení pro tuto továrnu.

Pokud se pokusíte spustit buď jiný port modulu listener, nebo jinou instanci EJB, musíte buď počkat na vrácení připojení do volného fondu připojení pro produkt `jms/CF1`.

Volné podprocesy údržby fondu připojení
Ke každému volnému fondu připojení je přidružen podproces údržby fondu, který monitoruje volný fond, aby se zajistilo, že připojení v něm jsou stále platná.

Pokud se podproces údržby fondu rozhodne, že připojení ve volném fondu musí být vyřazeno, podproces fyzicky zavře JMS připojení k IBM MQ.

Jak funguje podproces údržby fondu

Chování podprocesu údržby fondu je určeno hodnotou čtyř vlastností fondu připojení:

Časový limit stáří
Doba, po kterou zůstává připojení otevřené.

Minimální počet připojení
Minimální počet připojení, která správce připojení uchovává ve volném fondu továrny připojení.

Interval spuštění
Jak často se spouští podproces údržby fondu.

Nevyužitý časový limit
Jak dlouho zůstane připojení ve volném fondu, než bude uzavřeno.

Standardně se spravovaný podproces fondu spouští každých 180 sekund, i když tuto hodnotu lze změnit nastavením vlastnosti **Reap time** fondu připojení.

Podproces údržby zkontroluje každé připojení ve fondu, zkontroluje, jak dlouho bylo ve fondu a kolik času uplynulo od jeho vytvoření a posledního použití.

Pokud nebylo připojení použito po dobu delší, než je hodnota vlastnosti **Unused timeout** pro fond připojení, podproces údržby zkontroluje počet připojení, která jsou aktuálně ve fondu volných připojení. Pokud je toto číslo:

- Větší než hodnota **Minimum connections**, správce připojení uzavře připojení.
- Rovná se hodnotě **Minimum connections**, připojení není uzavřeno a zůstává ve volném fondu.

Výchozí hodnota vlastnosti **Minimum connections** je *1*, což znamená, že z důvodů výkonu se správce připojení vždy pokusí uchovat alespoň jedno připojení ve fondu volných prostředků.

Vlastnost **Unused timeout** má výchozí hodnotu 1800 sekund. Ve výchozím nastavení, pokud je připojení vloženo zpět do volného fondu a není znovu použito po dobu alespoň 1800 sekund, je toto připojení uzavřeno za předpokladu, že je uzavřeno, ponechá alespoň jedno připojení ve volném fondu.

Tento postup zabraňuje tomu, aby se nepoužívaná připojení stala zastaralými. Chcete-li tuto funkci vypnout, nastavte vlastnost **Unused timeout** na nulu.

Pokud je připojení ve volném fondu a uplynulá doba od jeho vytvoření je větší než hodnota vlastnosti **Aged timeout** pro fond připojení, pak je uzavřeno bez ohledu na to, jak dlouho bylo od posledního použití.

Standardně je vlastnost **Aged timeout** nastavena na nulu, což znamená, že podproces údržby tuto kontrolu nikdy neprovede. Připojení, která byla delší dobu než vlastnost **Aged timeout** , jsou vyřazena bez ohledu na to, kolik připojení zůstane ve volném fondu. Všimněte si, že vlastnost **Minimum connections** nemá v této situaci žádný vliv.

Zakázání podprocesu údržby fondu

Z předchozího popisu můžete vidět, že podproces údržby fondu provádí velký kus práce, když je aktivní, zejména pokud je ve volném fondu továrny připojení velký počet připojení.

Předpokládejme například, že existují tři továrny připojení JMS s vlastností **Maximum connections** nastavenou na 10 pro každou továrnu. Každých 180 sekund se tři podprocesy údržby fondu stanou aktivními a skenují volné fondy pro každou továrnu na připojení. Pokud mají volné fondy mnoho připojení, mají podprocesy údržby hodně práce, což může mít významný dopad na výkon.

Podproces údržby fondu můžete zakázat pro jednotlivý fond volných připojení nastavením jeho vlastnosti **Reap time** na nulu.

Zakázání podprocesu údržby znamená, že připojení nejsou nikdy zavřena, a to ani v případě, že uplynul **Unused timeout** . Avšak připojení mohou být i nadále zavřena, pokud byl parametr **Aged timeout** úspěšný.

Po dokončení aplikace s připojením správce připojení zkontroluje, jak dlouho připojení existovalo, a pokud je toto období delší než hodnota vlastnosti **Aged timeout** , správce připojení připojení uzavře připojení a nevrátí je do volného fondu.

Transakční důsledky vypršení časového limitu životnosti

Jak je popsáno v předchozí části, vlastnost **Aged timeout** určuje, jak dlouho zůstane připojení k poskytovateli JMS otevřené, než jej správce připojení zavře.

Výchozí hodnota vlastnosti **Aged timeout** je nula, což znamená, že připojení nebude nikdy uzavřeno, protože je příliš staré. Měli byste ponechat vlastnost **Aged timeout** na této hodnotě, protože povolení **Aged timeout** může mít transakční důsledky při použití JMS uvnitř EJB.

V systému JMSje jednotkou transakce JMS *relace*, která je vytvořena z JMS *připojení*. Jedná se o JMS *relaci* , která je zapsána do transakcí, a ne JMS *připojení*.

Vzhledem k návrhu aplikačního serveru lze připojení produktu JMS zavřít, protože uplynula doba **Aged timeout** , a to i v případě, že relace JMS vytvořené z tohoto připojení jsou zahrnuty do transakce.

Zavření připojení JMS způsobí odvolání veškeré nevyřízené transakční práce v relacích JMS, jak je popsáno ve specifikaci JMS . Aplikační server však neví, 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 JMS z EJB, ujistěte se, že veškerá práce JMS je explicitně potvrzena v relaci JMS před metodou EJB, která provádí uživatelské procedury operací JMS .

Příklady podprocesů údržby fondu
Pomocí příkladu Enterprise JavaBean (EJB) zjistěte, jak funguje podproces údržby fondu. Všimněte si, že můžete také použít objekty typu message-driven bean (MDB) a porty modulu listener, protože vše, co potřebujete, je způsob, jak získat připojení ve volném fondu.

Další podrobnosti o metodě `sendMessage()` viz "Způsob, jakým aplikace provádějící odchozí zasílání zpráv používají fond připojení" na stránce 299 .

Nakonfigurovali jste továrnu připojení s následujícími hodnotami:

- **`Reap time`** při výchozí hodnotě 180 sekund
- **`Aged timeout`** na výchozí hodnotě nula sekund
- **`Unused timeout`** nastavit na 300 sekund

Po spuštění aplikačního serveru je vyvolána metoda `sendMessage()` .

Tato metoda vytvoří připojení s názvem, například `c1`, používající továrnu `jms/CF1`, použije tuto továrnu k odeslání zprávy a poté zavolá `connection.close()`, což způsobí, že se `c1` vloží do volného fondu.

Po 180 sekundách se spustí podproces údržby fondu a podívá se na fond volných připojení `jms/CF1` . Ve fondu je volné připojení `c1` , takže podproces údržby se podívá na čas, kdy bylo připojení vráceně, a porovná jej s aktuálním časem.

Od doby, kdy bylo připojení vloženo do volného fondu, uplynulo 180 sekund, což je méně než hodnota vlastnosti **`Unused timeout`** pro `jms/CF1`. Proto závit údržby opouští samotné připojení.

O 180 sekund později se podproces údržby fondu spustí znovu. Podproces údržby vyhledá připojení `c1`a určí, že připojení bylo ve fondu po dobu 360 sekund, což je delší než nastavená hodnota **`Unused timeout`** , takže správce připojení připojení uzavře připojení.

Pokud nyní znovu spustíte metodu `sendMessage()` a aplikace zavolá `connectionFactory.createConnection()` , vytvoří správce připojení 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`** , aby zabránil zastaralému připojení, když je vlastnost **`Aged timeout`** nastavena na nulu.

Jak funguje vlastnost **`Aged timeout`** ?

V následujícím příkladu předpokládejme, že jste nastavili:

- Vlastnost **`Aged timeout`** na 300 sekund
- Vlastnost **`Unused timeout`** má hodnotu nula.

Vyvoláte 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, správce připojení vytvoří nové připojení `c1`a vrátí jej aplikaci. Když `sendMessage()` volá `connection.close()`, `c1` se vloží zpět do volného fondu připojení.

O 180 sekund později se spustí podproces údržby fondu. Podproces vyhledá `c1` ve fondu volných připojení a zkontroluje, jak dlouho byl vytvořen. Připojení existovalo po dobu 180 sekund, což je méně než **`Aged timeout`**, takže podproces údržby fondu jej ponechá sám a přejde zpět do režimu spánku.

O 60 sekund později je znovu volána funkce `sendMessage()` . Tentokrát, když metoda volá `connectionFactory.createConnection()`, správce připojení zjistí, že existuje připojení, `c1`, které je k dispozici ve volném fondu pro `jms/CF1`. Správce připojení vyjme produkt `c1` z volného fondu a předá toto připojení k aplikaci.

Připojení je vráceno do volného fondu při ukončení `sendMessage()` . O 120 sekund později se podproces údržby fondu znovu probudí, skenuje obsah volného fondu pro `jms/CF1` a zjišťuje `c1`.

Přestože bylo připojení použito pouze před 120 sekundami, podproces údržby fondu připojení zavře, protože připojení existovalo celkem 360 sekund, což je delší než hodnota 300 sekund, kterou jste nastavili pro vlastnost **`Aged timeout`** .

Jak vlastnost Minimální počet připojení ovlivňuje podproces údržby fondu

Při opětovném použití příkladu “Jak porty modulu listener MDB používají fond připojení” na stránce 298 předpokládejme, že máte na aplikačním serveru implementovány dva objekty MDB, z nichž každý používá jiný port modulu listener.

Každý port modulu listener je konfigurován tak, aby používal továrnu připojení `jms/CF1` , kterou jste nakonfigurovali pomocí:

- Vlastnost **`Unused timeout`** nastavena na 120 sekund
- Vlastnost **`Reap time`** nastavena na 180 sekund
- Vlastnost **`Minimum connections`** nastavena na hodnotu 1

Předpokládejme, že první modul listener je zastaven a jeho připojení `c1` je vloženo do volného fondu. O 180 sekund později se podproces údržby fondu probudí, skenuje obsah volného fondu pro produkt `jms/CF1`a zjistí, že produkt `c1` byl ve volném fondu déle než hodnota vlastnosti **`Unused timeout`** pro továrnu připojení.

Avšak před zavřením produktu `c1`se podproces údržby fondu podívá, kolik připojení zůstane ve fondu, pokud je toto připojení vyhozeno. Vzhledem k tomu, že `c1` je jediným připojením ve fondu volných připojení, správce připojení jej nezavře, protože by tím byl počet připojení, která zůstávají ve fondu volných připojení, menší než hodnota nastavená pro **`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. Do této doby byl produkt `c1` ve fondu volných připojení po dobu 360 sekund a produkt `c2` po dobu 180 sekund.

Podproces údržby fondu zkontroluje produkt `c1` a zjistí, že byl ve fondu déle než hodnota vlastnosti **`Unused timeout`** .

Podproces poté zkontroluje, kolik připojení je ve volném fondu, a porovná jej s hodnotou vlastnosti **`Minimum connections`** . Jelikož fond obsahuje dvě připojení a parametr **`Minimum connections`** je nastaven na hodnotu 1, správce připojení zavře `c1`.

Podproces údržby se nyní dívá na `c2`. Tato hodnota je také ve fondu volných připojení delší než hodnota vlastnosti **`Unused timeout`** . Avšak vzhledem k tomu, že zavření produktu `c2` by ponechalo volný fond připojení s méně než nastaveným minimálním počtem připojení v něm, správce připojení ponechá `c2` na pokoji.

JMS připojení a IBM MQ
Informace o použití produktu IBM MQ jako poskytovatele JMS .

Použití přenosu vazeb

Pokud byla továrna připojení nakonfigurována tak, aby používala přenos vazeb, každé připojení produktu JMS vytvoří konverzaci (označovanou také jako **hconn**) s produktem IBM MQ. Konverzace používá komunikaci mezi procesy (nebo sdílenou paměť) ke komunikaci se správcem front.

Použití přenosu klienta

Pokud byla továrna připojení poskytovatele systému zpráv IBM MQ nakonfigurována tak, aby používala transport klienta, každé připojení vytvořené z této továrny vytvoří novou konverzaci (označovanou také jako**hconn**) s produktem IBM MQ.

U továren připojení, které se připojují ke správci front pomocí normálního režimu poskytovatele systému zpráv IBM MQ , je možné, aby více připojení produktu JMS vytvořených z továrny připojení sdílelo připojení TCP/IP s produktem IBM MQ. Další informace viz “Sdílení připojení TCP/IP v produktu IBM MQ classes for JMS” na stránce 306.

Chcete-li určit maximální počet kanálů klienta, které připojení produktu JMS používají současně, přidejte hodnotu vlastnosti *Maximální počet připojení* pro všechny továrny připojení, které odkazují na stejného správce front.

Předpokládejme například, že máte dvě továrny připojení, `jms/CF1` a `jms/CF2` , které byly konfigurovány pro připojení ke stejnému správci front IBM MQ pomocí stejného kanálu IBM MQ .

Tyto továrny používají výchozí vlastnosti fondu připojení, což znamená, že hodnota *Maximální počet připojení* je nastavena na 10. Pokud se všechna připojení používají současně z jms/CF1 i z jms/CF2 , bude mezi aplikačním serverem a produktem IBM MQ20 konverzací.

Pokud se továrna připojení připojuje ke správci front pomocí normálního režimu poskytovatele systému zpráv IBM MQ , bude maximální počet připojení TCP/IP, která mohou existovat mezi aplikačním serverem a správcem front pro tyto továrny připojení, následující:

```
20/the value of SHARECNV for the IBM MQ channel
```

Pokud je továrna připojení konfigurována pro připojení pomocí režimu migrace poskytovatele systému zpráv IBM MQ , bude maximální počet připojení TCP/IP mezi aplikačním serverem a produktem IBM MQ pro tyto továrny připojení 20 (jedno pro každé připojení JMS ve fondech připojení pro obě továrny).

Související pojmy

“Použití IBM MQ classes for JMS/Jakarta Messaging” na stránce 78
IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging jsou poskytovatelé systému zpráv Java dodávaní s produktem IBM MQ. Kromě implementace rozhraní definovaných ve specifikacích JMS a Jakarta Messaging tito poskytovatelé systému zpráv přidávají do rozhraní API systému zpráv Java dvě sady rozšíření.

Sdružování objektů v prostředí Java SE
S produktem Java SE (nebo s jiným rámcem, jako je například Spring) jsou programovací modely extrémně flexibilní. Proto jednotná strategie sdružování nevyhovuje všem. Měli byste zvážit, zda existuje rámec, který by mohl provádět jakoukoli formu sdružování, například jaro.

Jinak by to mohla převzít logika aplikace. Zeptejte se sami sebe, jak složitá je aplikace sama o sobě? Je nejlepší porozumět aplikaci a tomu, co vyžaduje od konektivity k systému zpráv. Aplikace jsou často psány také v rámci vlastního kódu obálky kolem základního rozhraní JMS API.

I když to může být velmi rozumný přístup a může skrývat složitost, stojí za to mít na paměti, že může zavést problémy. Například generická metoda `getMessage()` , která je často volána, by neměla pouze otevírat a zavírat spotřebitele.

Body byste měli zvážit:

- Jak dlouho bude aplikace potřebovat přístup k produktu IBM MQ? Pořád, nebo jen občas.
- Jak často budou zprávy odesílány? Čím méně často, tím více může být jedno připojení k produktu IBM MQ sdíleno.
- Výjimka přerušeného připojení je obvykle známkou nutnosti znovu vytvořit připojení ve fondu. A co třeba:
  - Výjimky zabezpečení nebo hostitel není k dispozici
  - Výjimky zaplnění fronty
- Pokud dojde k výjimce přerušeného připojení, co by se mělo stát s ostatními volnými připojeními ve fondu? Měly by být uzavřeny a znovu vytvořeny?
- Používáte-li například protokol TLS, jak dlouho chcete, aby jedno připojení zůstalo otevřené?
- Způsob identifikace připojení ve fondu umožňuje administrátorovi správce front zjistit připojení a zpětně je sledovat.

Měli byste zvážit všechny objekty JMS pro použití fondu a tento objekt sdružit, kdykoli je to možné. Mezi objekty patří:

- JMS připojení

- Relace
- Kontexty
- Výrobci a spotřebitelé všech různých typů

Při použití přenosu klienta budou připojení, relace a kontexty produktu JMS používat sokety při komunikaci se správcem front produktu IBM MQ . Sdružením těchto objektů se ušetří počet příchozích připojení IBM MQ (`hConns`) ke správci front a sníží se počet instancí kanálu.

Použití přenosu vazeb do správce front zcela odebere síťovou vrstvu. Mnoho aplikací však používá přenos klienta k zajištění konfigurace s vyšší dostupností a vyrovnanou pracovní zátěží.

Producenti a spotřebitelé produktu JMS otevírají místa určení ve správci front. Pokud je otevřeno méně front nebo témat a tyto objekty používá více částí aplikace, může být jejich použití ve fondu užitečné.

Z perspektivy IBM MQ tento proces ukládá posloupnost operací MQOPEN a MQCLOSE.

Připojení, relace a kontexty

Všechny tyto objekty zapouzdřují manipulátory připojení IBM MQ ke správci front a jsou generovány ze serveru `ConnectionFactory`. Do aplikace můžete přidat logiku, která omezí počet připojení a dalších objektů vytvořených z jedné továrny na připojení na specifické číslo.

V aplikaci můžete použít jednoduchou datovou strukturu, která bude obsahovat vytvořená připojení. Kód aplikace, který potřebuje použít jednu z těchto datových struktur, může *zapůjčit* objekt, který se má použít.

Vezměte v úvahu následující faktory:

- Kdy mají být připojení odebrána z fondu? Obecně platí, že na připojení vytvořte modul listener pro výjimky. Je-li tento modul listener volán ke zpracování výjimky, měli byste znovu vytvořit připojení a všechny relace vytvořené z tohoto připojení.
- Pokud je tabulka CCDT používána pro vyrovnávání pracovní zátěže, mohou připojení přejít k různým správcům front. Tato volba může být použitelná pro požadavky na sdružování.

Pamatujte, že specifikace JMS uvádí, že se jedná o chybu programování pro více podprocesů, které přistupují k relaci nebo kontextu současně. Kód IBM MQ JMS se pokouší o důsledné zacházení s podprocesy. Měli byste však přidat do aplikace logiku, abyste zajistili, že relace nebo objekt kontextu bude v daném okamžiku používán pouze jedním podprocesem.

Výrobci a spotřebitelé

Každý vytvořený producent a spotřebitel otevře místo určení ve správci front. Bude-li stejné místo určení použito pro různé úlohy, má smysl ponechat objekty spotřebitele nebo producenta otevřené. Objekt zavřete pouze po dokončení veškeré práce.

Ačkoli otevření a zavření cíle jsou krátké operace, pokud jsou prováděny často, může čas potřebný k sečíst.

Rozsah těchto objektů je v rámci relace nebo kontextu, ze kterého jsou vytvořeny, a proto je třeba je v tomto rozsahu zadržovat. Obecně platí, že žádosti jsou napsány tak, aby to bylo docela jednoduché.

Monitorování

Jak budou aplikace monitorovat své fondy objektů? Odpověď na to je do značné míry určena složitostí řešení sdružování realizovaných.

Pokud uvažujete o implementaci fondu JavaEE, existuje velký počet voleb, včetně následujících:

- Aktuální velikost fondů
- Čas, který v nich objekty strávily
- Čištění bazénů
- Aktualizace připojení

Měli byste také zvážit, jak se ve správci front zobrazí jedna znovu použitá relace. Existují vlastnosti továrny připojení pro identifikaci aplikace (například appName), které mohou být užitečné.

"Použití IBM MQ classes for JMS/Jakarta Messaging" na stránce 78
IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging jsou poskytovatelé systému zpráv Java dodávaní s produktem IBM MQ. Kromě implementace rozhraní definovaných ve specifikacích JMS a Jakarta Messaging tito poskytovatelé systému zpráv přidávají do rozhraní API systému zpráv Java dvě sady rozšíření.

Sdílení připojení TCP/IP v produktu IBM MQ classes for JMS
Pro sdílení jednoho připojení TCP/IP lze vytvořit více instancí kanálu MQI.

Aplikace, které jsou spuštěny ve stejném běhovém prostředí Java a které používají adaptér prostředků IBM MQ classes for JMS nebo IBM MQ pro připojení ke správci front pomocí přenosu CLIENT, lze vytvořit pro sdílení instance kanálu.

Je-li kanál definován s parametrem **SHARECNV** nastaveným na hodnotu větší než 1, může tento počet konverzací sdílet instanci kanálu. Chcete-li povolit továrnu připojení nebo specifikaci aktivace pro použití této funkce, nastavte vlastnost **SHARECONVALLOWED** na hodnotu YES.

Každé JMS připojení a JMS relace vytvořené aplikací JMS vytvoří svou vlastní konverzaci se správcem front.

Při spuštění specifikace aktivace adaptér prostředků IBM MQ zahájí konverzaci se správcem front pro použití specifikace aktivace. Každá relace serveru ve fondu relací serveru, který je přidružen ke specifikaci aktivace, také spustí konverzaci se správcem front.

Atribut **SHARECNV** představuje nejlepší přístup ke sdílení připojení. Proto, když je hodnota **SHARECNV** větší než 0 použita s IBM MQ classes for JMS, není zaručeno, že nový požadavek na připojení vždy sdílí již zavedené připojení.

Způsob sdílení připojení TCP/IP

Pro sdílení připojení TCP/IP jsou k dispozici dvě strategie:

Strategie GLOBAL
Tato strategie je výchozí strategií pro sdílení připojení TCP/IP. Jakékoli připojení JMS nebo relace mohou používat konverzaci na libovolném vhodném připojení TCP/IP. Vhodnost je určena takovými faktory, jako je adresa hostitele, číslo portu, ID uživatele a heslo a parametry TLS/SSL.

Tento přístup pro sdílení připojení TCP/IP minimalizuje počet instancí kanálu, které jsou používány, ale za cenu soupeření o přístup ke globálnímu fondu připojení TCP/IP.

Strategie CONNECTION
Při použití této strategie jsou instance kanálu sdíleny pouze mezi souvisejícími objekty JMS . Při vytvoření připojení JMS je pro něj vytvořena instance kanálu a další konverzace na této instanci kanálu jsou k dispozici pouze pro relace JMS , které jsou vytvořeny tímto připojením JMS .

Pokud je vytvořeno více konverzací, než určuje atribut SHARECNV, vytvoří se nová instance kanálu, kterou mohou používat pouze relace JMS vytvořené původním připojením JMS .

Tento přístup ke sdílení instancí kanálu snižuje soupeření o konverzace na úkor potenciálně vyžadujících výrazně více instancí kanálu.

Explicitní určení strategie sdílení instance kanálu

V 9.3.2

Standardně se strategie GLOBAL používá v případě, že aplikace nelze znovu připojit. Znovu připojitelné aplikace vždy používají strategii CONNECTION.

Pro aplikace, které používají IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging, může být povolena strategie CONNECTION pro neopětovně připojitelné aplikace na úrovni celé aplikace. Strategii CONNECTION můžete povolit nastavením systémové vlastnosti

`com.ibm.mq.jms.channel.sharing` na hodnotu `CONNECTION`. Tato hodnota nerozlišuje velikost písmen a jakákoli jiná hodnota než `CONNECTION` se ignoruje.

Systémovou vlastnost `com.ibm.mq.jms.channel.sharing` můžete nastavit jedním z následujících způsobů:

- Nastavte vlastnost jako součást inicializace prostředí JVM pomocí volby příkazového řádku "-D":

```
-Dcom.ibm.mq.jms.channel.sharing=CONNECTION
```

- Nastavte vlastnost před použitím IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging pomocí `System.setProperty()`

Výpočet počtu instancí kanálu pro strategii sdílení GLOBAL

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 používá specifikace aktivace.

Další aplikace JMS

Počet instancí kanálu = (*jms_connections* + *jms_sessions*)/ *SHARECNV_value*

Kde *jms_connections* je počet připojení, která aplikace vytvořila, *jms_sessions* je počet relací JMS vytvořených aplikací a *SHARECNV_value* je hodnota vlastnosti **SHARECNV** na kanálu, který používá specifikace aktivace.

Výpočet počtu instancí kanálu pro strategii sdílení CONNECTION

Počet instancí kanálu závisí na distribuci relací JMS mezi připojeními JMS v aplikaci.

Povolte jednu konverzaci pro připojení JMS a jednu konverzaci pro každou relaci JMS pod tímto připojením JMS , pak vydělte hodnotou **SHARECNV** a zaokrouhlte nahoru. Tento výpočet poskytuje instance kanálu, které jsou potřebné pro dané připojení JMS .

Stejný princip lze použít pro specifikace aktivace. Považte specifikaci aktivace za připojení JMS a vlastnost **maxPoolDepth** za počet relací JMS .

Příklady

Následující příklady ukazují, jak použít vzorce pro výpočet počtu instancí kanálu, které jsou vytvořeny ve správci front aplikacemi pomocí adaptéru prostředků IBM MQ classes for JMS nebo IBM MQ .

Příklad aplikace JMS
Připojení aplikace JMS se připojí ke správci front pomocí přenosu CLIENT a vytvoří JMS připojení a tři JMS relace. 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 připojí ke správci front pomocí přenosu CLIENT. Specifikace aktivace je konfigurována s vlastností **maxPoolDepth** nastavenou na hodnotu 10. Kanál, pro jehož použití je specifikace aktivace konfigurována, má vlastnost **SHARECNV** nastavenu na hodnotu 10. Při spuštění specifikace aktivace a souběžném zpracování 10 zpráv je počet konverzací mezi specifikací aktivace a správcem front 11 (10 konverzací pro relace serveru a 1 pro specifikaci aktivace). Počet instancí kanálu používaných specifikací aktivace je 2.

Příklad specifikace aktivace
Specifikace aktivace se připojí ke správci front pomocí přenosu CLIENT. Specifikace aktivace je konfigurována s vlastností **maxPoolDepth** nastavenou na hodnotu 5. Kanál, pro jehož použití je

specifikace aktivace konfigurována, má vlastnost **SHARECNV** nastavenu na hodnotu 0. Při spuštění specifikace aktivace a souběžném zpracování 5 zpráv je počet konverzací mezi specifikací aktivace a správcem front 6 (pět konverzací pro relace serveru a jedna pro specifikaci aktivace). Počet instancí kanálu používaných specifikací aktivace je 6, protože vlastnost **SHARECNV** v kanálu je nastavena na hodnotu 0, každá konverzace používá svou vlastní instanci kanálu.

Související úlohy

“Určení počtu připojení TCP/IP vytvořených z WebSphere Application Server na IBM MQ” na stránce 488
Pomocí funkce sdílení konverzací 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
Pomocí vlastnosti LOCALADDRESS určete rozsah portů, ke kterým se může vaše aplikace vázat.

Když se aplikace IBM MQ classes for JMS pokusí připojit ke správci front IBM MQ v režimu klienta, brána firewall může povolit pouze ta připojení, která pocházejí ze zadaných portů nebo z řady portů. V této situaci můžete použít vlastnost LOCALADDRESS objektu ConnectionFactory, QueueConnectionFactory nebo TopicConnectionFactory k určení portu nebo rozsahu portů, ke kterým se může aplikace vázat.

Vlastnost LOCALADDRESS můžete nastavit pomocí nástroje pro administraci IBM MQ JMS nebo voláním metody setLocalAddress () v aplikaci JMS . Zde je příklad nastavení vlastnosti z aplikace:

```
mqConnectionFactory.setLocalAddress("192.0.2.0(2000,3000)");
```

Když se aplikace následně připojí ke správci front, připojí se k lokální adrese IP a číslu portu v rozsahu 192.0.2.0(2000) až 192.0.2.0(3000).

V systému s více než jedním síťovým rozhraním můžete také pomocí vlastnosti LOCALADDRESS určit, které síťové rozhraní musí být pro připojení použito.

Pro připojení zprostředkovatele v reálném čase je vlastnost LOCALADDRESS relevantní pouze při použití výběrového vysílání. V tomto případě můžete pomocí vlastnosti určit, které lokální síťové rozhraní musí být pro připojení použito, ale hodnota vlastnosti nesmí obsahovat číslo portu nebo rozsah čísel portů.

Pokud omezíte rozsah portů, mohou se vyskytnout chyby připojení. Dojde-li k chybě, dojde k výjimce JMSException s vloženou výjimkou MQException, která obsahuje kód příčiny IBM MQ MQRC_Q_MGR_NOT_AVAILABLE a následující zprávu:

```
Pokus o připojení soketu byl odmítnut kvůli omezením LOCAL_ADDRESS_PROPERTY
```

K chybě může dojít, pokud jsou všechny porty v uvedeném rozsahu používány, nebo pokud je zadaná adresa IP, název hostitele nebo číslo portu neplatné (například záporné číslo portu).

Vzhledem k tomu, že produkt IBM MQ classes for JMS může vytvářet jiná připojení než ta, která vyžaduje aplikace, vždy zvažte zadání rozsahu portů. Obecně platí, že každá relace vytvořená aplikací vyžaduje jeden port a produkt IBM MQ classes for JMS může vyžadovat tři nebo čtyři další porty. Pokud dojde k chybě připojení, zvyšte rozsah portů.

Sdružování připojení, které se standardně používá v produktu IBM MQ classes for JMS, může mít vliv na rychlost opětovného použití portů. V důsledku toho může během uvolnění portů dojít k chybě připojení.

Komprese kanálů v souboru IBM MQ classes for JMS
Aplikace IBM MQ classes for JMS může použít prostředky IBM MQ ke kompresi záhlaví nebo dat zprávy.

Komprese dat, která procházejí kanálem IBM MQ , může zlepšit výkon kanálu a snížit provoz na síti. Pomocí funkce dodávané s produktem IBM MQmůžete komprimovat data, která proudí kanály zpráv a kanály MQI. Na obou typech kanálu můžete komprimovat data záhlaví a data zprávy nezávisle na sobě. Při výchozím nastavení nejsou na kanálu komprimována žádná data.

Aplikace IBM MQ classes for JMS určuje techniky, které lze použít pro kompresi dat záhlaví nebo zprávy v připojení vytvořením objektu java.util.Collection . Každá technika komprese je celočíselný objekt v kolekci a pořadí, ve kterém aplikace přidává techniky komprese do kolekce, je pořadí, ve kterém jsou techniky komprese vyjednávány se správcem front, když aplikace vytváří připojení. Aplikace pak může

předat kolekci do objektu ConnectionFactory voláním metody setHdrCompList() pro data záhlaví nebo metody setMsgCompList() pro data zprávy. Když je aplikace připravena, může vytvořit připojení.

Následující fragmenty kódu ilustrují 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 zpráv:

```
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 vyjednány v pořadí RLE, pak ZLIBHIGH, když je vytvořeno připojení. Zvolenou techniku komprese nelze změnit během doby životnosti objektu připojení. Chcete-li použít kompresi pro připojení, musí být před vytvořením objektu připojení volána metoda setHdrCompList() a setMsgCompList().

Asynchronní vkládání zpráv do adresáře IBM MQ classes for JMS
Za normálních okolností, když aplikace odesílá zprávy do místa určení, musí aplikace čekat na správce front, aby potvrdila, že požadavek zpracovala. Za určitých okolností můžete zlepšit výkon systému zpráv volbou asynchronního vložení zpráv. Když aplikace vloží zprávu asynchronně, správce front nevrátí úspěch nebo selhání každého volání, ale místo toho můžete pravidelně kontrolovat chyby.

To, zda cíl vrátí řízení aplikaci bez určení, zda správce front přijal zprávu bezpečně, závisí na následujících vlastnostech:

JMS Cílová vlastnost PUTASYNCALLOWED (krátký název-PAALD).
PUTASYNCALLOWED řídí, zda aplikace JMS mohou vkládat zprávy asynchronně, pokud je tato volba povolena základní frontou nebo tématem, které představuje cíl JMS .

Vlastnost fronty nebo tématu IBM MQ DEFPRESP (výchozí typ odezvy vložení).
Parametr DEFPRESP určuje, zda aplikace, které vkládají zprávy do fronty nebo publikují zprávy do tématu, mohou používat funkci asynchronního vložení.

V následující tabulce jsou uvedeny možné hodnoty vlastností PUTASYNCALLOWED a DEFPRESP a kombinace hodnot používaných k povolení funkce asynchronního vložení:

Tabulka 46. Jak se kombinují vlastnosti PUTASYNCALLOWED a DEFPRESP, aby se zjistilo, zda jsou zprávy asynchronně vkládány do místa určení.

Vlastnost fronty IBM MQ	PUTASYNCALLOWED = NO	PUTASYNCALLOWED = ANO	Funkce asynchronního vložení povolena
DEFPRESP=SYNCHRONIZACE	Funkce asynchronního vložení není povolena	Funkce asynchronního vložení povolena	PUTASYNCALLOWED = AS_DEST nebo AS_Q_DEF nebo AS_T_DEF

Tabulka 46. Jak se kombinují vlastnosti PUTASYNCALLOWED a DEFPRESP, aby se zjistilo, zda jsou zprávy asynchronně vkládány do místa určení. (pokračování)

Vlastnost fronty IBM MQ	PUTASYNCALLOWED = NO	PUTASYNCALLOWED = ANO	Funkce asynchronního vložení povolena
DEFPRESP=ASYNC	Funkce asynchronního vložení není povolena	Funkce asynchronního vložení povolena	PUTASYNCALLOWED = AS_DEST nebo AS_Q_DEF nebo AS_T_DEF

Chování můžete změnit zadáním vlastnosti cíle IBM MQ-JMS na hodnotu "NO" nebo "YES", jak je uvedeno v tabulce, ale lze ji také přepsat pro celý virtuální počítač Java pomocí prostředí JVM **SystemProperty** a hodnoty:

```
com.ibm.mq.cfg.Channels.Put1DefaultAlwaysSync=Y
```

Pro zprávy odeslané v relaci transakce aplikace nakonec určí, zda správce front přijal zprávy bezpečně při volání `commit()`.

Pokud aplikace odešle trvalé zprávy v rámci relace s transakcemi a jedna nebo více zpráv nejsou přijaty bezpečně, transakce se nezdaří potvrdit a vygeneruje výjimku. Pokud však aplikace odesílá přechodné zprávy v rámci relace s transakcemi a jedna nebo více zpráv nejsou přijímány bezpečně, transakce se úspěšně potvrdí. Aplikace neobdrží žádnou zpětnou vazbu, že přechodné zprávy nebyly doručovány bezpečně.

Pro přechodné zprávy odeslané v relaci, která není transakční, určuje vlastnost SENDCHECKCOUNT objektu *ConnectionFactory* počet zpráv, které mají být odeslány, než produkt IBM MQ classes for JMS zkontroluje, zda správce front zprávy bezpečně přijal.

Pokud kontrola zjistí, že jedna nebo více zpráv nebylo přijato bezpečně, a aplikace zaregistrovala modul listener pro výjimky s připojením, IBM MQ classes for JMS zavolá metodu `onException()` modulu listener pro výjimky, aby aplikaci předala výjimku JMS .

Výjimka JMS má kód chyby JMSWMQ0028 a tento kód zobrazuje následující zprávu:

```
At least one asynchronous put message failed or gave a warning.
```

Výjimka JMS má také propojenou výjimku, která poskytuje další podrobnosti. Výchozí hodnota vlastnosti SENDCHECKCOUNT je nula, což znamená, že se žádné takové kontroly neprovádějí.

Tato optimalizace je nejpřínosnější 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 může tuto optimalizaci nadále používat i v případě, že se připojí ke správci front v režimu vazeb, ale očekávaný přínos výkonu není tak velký.

Poznámka: Používáte-li k odeslání zprávy v rámci transakce neidentifikovaný soubor **MessageProducer** , jsou zprávy standardně vkládány do fronty pomocí mechanizmu asynchronního vložení.

K tomu může dojít, protože rozhraní API JMS umožňuje vytvoření souboru **MessageProducer** bez určení cíle pomocí syntaxe:

```
javax.jms.MessageProducer messageProducer =  javax.jms.Session.createProducer(null);
  messageProducer.send(Destination destination, Message message, int deliveryMode, int priority, long
timeToLive);
```

V tomto scénáři je cíl JMS poskytnut, když je zpráva odeslána, spíše než předem, když je sestaven produkt **MessageProducer** . Pokud jde o rozhraní API IBM MQ , výsledkem je vydání příkazu MQPUT1 pro vložení zprávy do fronty.

Provedete-li to v synchronizačním bodu IBM MQ , což znamená (v terminologii JMS) vložení zprávy do transakce, buď pomocí transakční relace JMS , nebo pomocí XASession přepínače rozhraní API IBM MQ classes for JMS pro použití asynchronního vložení.

Použití dopředného čtení s IBM MQ classes for JMS
Funkce dopředného čtení, kterou poskytuje produkt IBM MQ , umožňuje, aby byly přechodné zprávy přijaté mimo transakci odeslány do produktu IBM MQ classes for JMS před tím, než si je aplikace vyžádá. Produkt IBM MQ classes for JMS ukládá zprávy do interní vyrovnávací paměti a předává je aplikaci, když o ně aplikace požádá.

Aplikace systému IBM MQ classes for JMS , které používají MessageConsumers nebo MessageListeners pro příjem zpráv z místa určení mimo transakci, mohou používat funkci dopředného čtení. Použití dopředného čtení umožňuje aplikacím, které používají tyto objekty, těžit ze zlepšeného výkonu při přijímání zpráv.

To, zda aplikace používající MessageConsumers nebo MessageListeners může používat dopředné čtení, závisí na následujících vlastnostech:

Cílová vlastnost JMS READAHEADALLOWED (krátký název-RAALD).
Volba READAHEADALLOWED určuje, zda mohou aplikace JMS při získávání nebo procházení dočasných zpráv mimo transakci používat dopředné čtení, pokud to umožňuje základní fronta nebo téma reprezentované místem určení JMS .

Vlastnost fronty nebo tématu IBM MQ DEFREADA (výchozí dopředné čtení).
Program DEFREADA určuje, zda aplikace, které přijímají nebo procházejí dočasné zprávy mimo transakci, mohou používat dopředné čtení.

V následující tabulce jsou uvedeny možné hodnoty vlastností READAHEADALLOWED a DEFREADA a kombinace hodnot, které umožňují funkci dopředného čtení:

Tabulka 47. Jak se vlastnosti READAHEADALLOWED a DEFREADA kombinují, aby se zjistilo, zda se při příjmu nebo procházení dočasných zpráv mimo transakci používá dopředné čtení.

Vlastnost fronty IBM MQ	**READAHEADALLOWED = ANO**	**READAHEADALLOWED = NE**	**AS_DEST nebo AS_Q_DEF nebo AS_T_DEF**
DEFREADA = NE	Funkce dopředného čtení povolena	Funkce dopředného čtení není povolena	Funkce dopředného čtení není povolena
DEFREADA = ANO	Funkce dopředného čtení povolena	Funkce dopředného čtení není povolena	Funkce dopředného čtení povolena
DEFREADA = ZAKÁZÁNO	Funkce dopředného čtení není povolena	Funkce dopředného čtení není povolena	Funkce dopředného čtení není povolena

Je-li povolena funkčnost dopředného čtení, když aplikace vytvoří MessageConsumer nebo MessageListener , vytvoří IBM MQ classes for JMS vnitřní vyrovnávací paměť pro místo určení, které MessageConsumer nebo MessageListener monitoruje. Pro každý MessageConsumer nebo MessageListenerexistuje jedna interní vyrovnávací paměť. Správce front začne odesílat dočasné zprávy do produktu IBM MQ classes for JMS , když aplikace volá jednu z následujících metod:

- MessageConsumer.receive()
- MessageConsumer.receive(long timeout)
- MessageConsumer.receiveNoWait()
- Session.setMessageListener(MessageListener listener)

IBM MQ classes for JMS automaticky vrátí první zprávu zpět do aplikace pomocí volání metody, které aplikace provedla. Ostatní dočasné zprávy jsou uloženy produktem IBM MQ classes for JMS ve vnitřní vyrovnávací paměti, která byla vytvořena pro dané místo určení. Když aplikace vyžádá další zprávu ke zpracování, IBM MQ classes for JMS vrátí další zprávu ve vnitřní vyrovnávací paměti.

Pokud je interní vyrovnávací paměť prázdná, produkt IBM MQ classes for JMS od správce front požaduje další přechodné zprávy.

Vnitřní vyrovnávací paměť, kterou používá IBM MQ classes for JMS , se odstraní, když aplikace zavře `MessageConsumer`nebo JMS relaci, ke které je přidružen `MessageListener` .

V případě systému `MessageConsumers`jsou všechny nezpracované zprávy ve vnitřní vyrovnávací paměti ztraceny.

Při použití `MessageListeners`závisí to, co se stane se zprávami ve vnitřní vyrovnávací paměti, na cílové vlastnosti JMS READAHEADCLOSEPOLICY (krátký název-RACP). Výchozí hodnota vlastnosti je DELIVER_ALL, což znamená, že relace JMS , která byla použita k vytvoření souboru `MessageListener` , nebude uzavřena, dokud nebudou aplikaci doručeny všechny zprávy v interní vyrovnávací paměti. Je-li vlastnost nastavena na hodnotu DELIVER_CURRENT, bude relace JMS po zpracování aktuální zprávy aplikací uzavřena a všechny zbývající zprávy v interní vyrovnávací paměti budou zrušeny.

Zachovaná publikování v produktu IBM MQ classes for JMS
Klienta IBM MQ classes for JMS lze nakonfigurovat tak, aby používal zachovaná publikování.

Vydavatel může určit, že kopie publikování musí být zachována, aby mohla být odeslána budoucím odběratelům, kteří zaregistrují zájem o dané téma. To se provádí v produktu IBM MQ classes for JMS nastavením celočíselné vlastnosti JMS_IBM_RETAIN na hodnotu 1. Konstanty byly definovány pro tyto hodnoty v rozhraní com.ibm.msg.client.jms.JmsConstants . Pokud jste například vytvořili zprávu *msg*, nastavte ji jako zachované publikování pomocí následujícího kódu:

```
// set as a retained publication
msg.setIntProperty(JmsConstants.JMS_IBM_RETAIN, JmsConstants.RETAIN_PUBLICATION);
```

Nyní můžete zprávu odeslat jako normální. JMS_IBM_RETAIN lze také dotazovat v přijaté zprávě. Proto je možné se dotazovat, zda je přijatá zpráva zachované publikování.

Podpora XA v produktu IBM MQ classes for JMS
Produkt JMS podporuje transakce vyhovující standardu XA v režimech vazeb a klientů s podporovaným správcem transakcí v rámci kontejneru JEE .

Pokud vyžadujete funkčnost XA v prostředí aplikačního serveru, musíte aplikaci nakonfigurovat odpovídajícím způsobem. Informace o tom, jak konfigurovat aplikace pro použití distribuovaných transakcí, naleznete ve vlastní dokumentaci aplikačního serveru.

Správce front IBM MQ nemůže fungovat jako správce transakcí pro produkt JMS.

Prodleva doručení pro zprávy JMS
V případě systému JMS 2.0 nebo novějšího můžete při odesílání zprávy určit prodlevu doručení. Správce front zprávu nedoručí, dokud neuplyne určená prodleva doručení.

Aplikace může uvést prodlevu doručení v milisekundách, když odešle zprávu, buď pomocí `MessageProducer.setDeliveryDelay(long deliveryDelay)` , nebo `JMSProducer.setDeliveryDelay(long deliveryDelay)`. Tato hodnota se přičítá k času, kdy je zpráva odeslána, a poskytuje nejdřívější čas, kdy může jiná aplikace tuto zprávu získat.

Prodleva doručení je implementována pomocí jedné interní fronty fázování. 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 prodlevy doručení, monitoruje zprávy ve frontě fázování. Po dokončení prodlevy doručení zprávy je zpráva převzata z pracovní fronty a umístěna do cílové fronty.

Klienti systému zpráv

Implementace IBM MQ prodlevy doručení je k dispozici pouze v případě, že používáte klienta JMS . Následující omezení platí, pokud používáte prodlevu doručení s produktem IBM MQ. Tato omezení platí stejně pro `MessageProducers` a `JMSProducers`, ale v případě `JMSProducers`jsou vyvolána `JMSRuntimeExceptions` .

- Jakýkoli pokus o volání funkce `MessageProducer.setDeliveryDelay` s nenulovou hodnotou při připojení ke správci front staršímu než IBM MQ 8.0bude mít za následek zprávu `JMSException` se zprávou MQRC_FUNCTION_NOT_SUPPORTED.
- Prodleva doručení není podporována pro klastrovaná místa určení, která mají jinou hodnotu **DEFBIND** než MQBND_BIND_NOT_FIXED. Pokud má `MessageProducer` nastavenou nenulovou prodlevu doručení a je učiněn pokus o odeslání do místa určení, které nesplňuje tento požadavek, pak volání vyústí ve zprávu `JMSException` se zprávou MQRC_OPTIONS_ERROR.
- Jakýkoli pokus o nastavení hodnoty doby platnosti, která je menší než dříve určená nenulová prodleva doručení, nebo naopak, bude mít za následek `JMSException` se zprávou MQRC_EXPIRY_ERROR. Tato kontrola se provádí při volání metod `setTimeToLive` nebo `setDeliveryDelay` nebo `send` v závislosti na přesné sadě zvolených operací.
- Použití zachovaných publikování a prodlevy doručení nejsou podporovány. Pokus o publikování zprávy s prodlevou doručení, pokud byla tato zpráva označena jako uchovaná pomocí produktu `msg.setIntProperty(JmsConstants.JMS_IBM_RETAIN, JmsConstants.RETAIN_PUBLICATION)` , bude mít za následek `JMSException` se zprávou MQRC_OPTIONS_ERROR.
- Prodleva doručení a seskupování zpráv není podporováno a jakýkoli pokus o použití této kombinace má za následek `JMSException` se zprávou MQRC_OPTIONS_ERROR.

Jakékoli selhání odeslání zprávy s prodlevou doručení má za následek, že klient vygeneruje zprávu `JMSException` s vhodnou chybovou zprávou, například fronta je plná. V některých situacích se může chybová zpráva vztahovat na cílové místo určení, frontu fázování nebo obojí.

Poznámka: Produkt IBM MQ umožňuje aplikacím, které vložily zprávu do pracovní jednotky, znovu získat stejnou zprávu, i když pracovní jednotka nebyla potvrzena. Tato technika nefunguje s prodlevou doručení, protože zpráva není umístěna do fronty fázování, dokud není transakce potvrzena, a v důsledku toho nebude odeslána do cílového místa určení.

Autorizace

Produkt IBM MQ provádí kontroly autorizace původního cílového místa určení, když aplikace odešle zprávu s nenulovým zpožděním doručení. Není-li aplikace autorizována, odeslání se nezdaří. Když správce front zjistí, že byla prodleva doručení zprávy dokončena, otevře cílovou frontu. V tomto bodě se neprovádějí žádné kontroly autorizace.

SYSTEM.DDELAY.LOCAL.QUEUE

Systémová fronta, SYSTEM.DDELAY.LOCAL.QUEUEse používá k implementaci prodlevy doručení.

- Multi V systému Multiplatforms, SYSTEM.DDELAY.LOCAL.QUEUE existuje standardně. Systémová fronta musí být změněna tak, aby její atributy MAXMSGL a MAXDEPTH byly dostatečné pro očekávanou zátěž.
- z/OS V systému IBM MQ for z/OS, SYSTEM.DDELAY.LOCAL.QUEUE se používá jako fronta fázování pro zprávy odeslané s prodlevou doručení do lokálních i sdílených front. V systému z/OSmusí být fronta vytvořena a definována tak, aby její atributy MAXMSGL a MAXDEPTH byly dostatečné pro očekávané načtení.

Když je tato fronta vytvořena, musí být zabezpečena tak, aby k ní mělo přístup co nejméně uživatelů. Přístup k frontě musí být pouze pro účely údržby a monitorování.

Když je zpráva odeslána aplikací JMS s nenulovým zpožděním doručení, je vložena do této fronty s novým ID zprávy. Původní ID zprávy je umístěno 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

Pokud je váš systém spuštěn na systému z/OS, je třeba vzít v úvahu další aspekty, které je třeba vzít v úvahu, pokud chcete použít zpoždění doručení.

Pokud se má použít prodleva doručení, systémová fronta SYSTEM.DDELAY.LOCAL.QUEUE musí být definována. Musí být definována s paměťovou třídou, která je dostatečná pro očekávanou zátěž, a s uvedeným parametrem INDXTYPE (NONE) a MSGDLVSQ (FIFO). V JCL CSQ4INSG je poskytnuta ukázková definice systémové fronty, která je označena jako komentář.

Sdílené fronty

Prodleva doručení je podporována pro odesílání zpráv do sdílených front. Existuje však pouze jedna soukromá fronta fázování, která se používá bez ohledu na to, zda je cílová fronta sdílená či nikoli. Správce front, který vlastní tuto soukromou frontu, musí být spuštěn, aby po dokončení prodlevy odeslal zpožděnou zprávu do cílové sdílené fronty.

Poznámka: Je-li do sdílené fronty vložena dočasná zpráva s prodlevou doručení a správce front, který je vlastníkem fronty fázování, se vypne, původní zpráva bude ztracena. Výsledkem je, že přechodné zprávy odeslané se zpožděním doručení do sdílené fronty budou pravděpodobně ztraceny více než přechodné zprávy odeslané bez prodlevy doručení do sdílené fronty.

Rozlišení cílového místa určení

Je-li zpráva odeslána do fronty, je rozlišení řízeno dvakrát; jednou aplikací JMS a jednou správcem front, když zprávu vyřadí z fronty fázování a odešle ji do cílové fronty.

Cílové odběry pro publikování se shodují, když aplikace JMS volá metodu odeslání.

Pokud je zpráva odeslána s perzistencí nebo prioritou podle definice fronty, pak je hodnota nastavena v prvním rozlišení a ne v druhém.

Interval vypršení

Prodleva doručení zachovává chování vlastnosti vypršení platnosti, **MQMD.Expiry**. Pokud byla například vložena zpráva z aplikace JMS s intervalem vypršení platnosti 20 000 ms a prodlevou doručení 5 000 ms a byla přijata po uplynulé době 10 000 ms, hodnota pole MQMD.expiry může být přibližně 50 desetin sekundy. Tato hodnota označuje, že uplynulo 15 sekund od doby, kdy byla zpráva vložena, do doby, kdy byla přijata.

Pokud dojde k vypršení platnosti zprávy ve frontě fázování a je nastavena jedna z voleb MQRO_EXPIRATION_ *, bude vygenerovaná sestava určena pro původní zprávu odeslanou aplikací. Záhlaví, které obsahuje informace o prodlevě doručení, bude odebráno.

Zastavení a spuštění procesoru prodlevy doručení

z/OS V systému z/OSje procesor prodlevy doručení integrován do adresního prostoru MSTR správce front. Při spuštění správce front se spustí také procesor prodlevy doručení. Je-li fronta dočasného uložení objektů k dispozici, otevře frontu a čeká na doručení zpráv ke zpracování. Pokud fronta fázování nebyla definována nebo je zakázána pro získání nebo dojde k jiné chybě, procesor prodlevy doručení se vypne. Je-li fronta fázování později definována nebo změněna tak, aby byla povolena, procesor prodlevy doručení se restartuje. Pokud se procesor prodlevy doručení vypne z jiného důvodu, lze jej restartovat změnou atributu **PUT** fronty fázování z hodnoty POVOLENO na hodnotu ZAKÁZÁNO a znovu na hodnotu POVOLENO . Chcete-li z libovolné příčiny zastavit procesor prodlevy doručení, nastavte atribut PUT fronty fázování na hodnotu DISABLED.

Multi V systému Multiplatformsse procesor prodlevy spustí se správcem front a automaticky se restartuje v případě zotavitelného selhání.

Nepodařilo se vložit do cílové fronty

Pokud po dokončení prodlevy nelze zpožděnou zprávu vložit do cílové fronty, je zpráva řešena tak, jak je uvedeno v jejích volbách sestavy: je buď vyřazena, nebo odeslána do fronty nedoručených zpráv. Pokud se tato akce nezdaří, dojde k pokusu o pozdější vložení zprávy. Pokud je akce úspěšná, vygeneruje se zpráva o výjimce a odešle se do uvedené fronty, pokud je sestava požadována. Pokud zprávu sestavy nelze odeslat, zpráva sestavy se odešle do fronty nedoručených zpráv. Pokud se odeslání sestavy do fronty nedoručených zpráv nezdaří a zpráva je trvalá, všechny změny se vyřadí a původní zpráva se odvolá a znovu doručí později. Pokud je zpráva dočasná, zpráva sestavy je vyřazena, ale jiné změny jsou potvrzeny. Pokud nelze zpožděné publikování doručit, protože odběratel zrušil odběr, nebo v případě netrvalého odběratele, protože se odpojil, je zpráva bezobslužně zrušena. Zprávy sestavy jsou stále generovány tak, jak je popsáno dříve.

Pokud zpožděné publikování nelze doručit odběrateli a je místo toho vloženo do fronty nedoručených zpráv a vložení do fronty nedoručených zpráv se nezdaří, zpráva se zahodí.

Chcete-li snížit pravděpodobnost selhání operace vložení do cílové fronty po dokončení prodlevy doručení, správce front provede některé základní kontroly v případě, že klient produktu JMS odešle zprávu s nenulovou prodlevou doručení. Tyto kontroly zahrnují, zda je fronta vypnuta, zda je zpráva větší než maximální povolená délka zprávy a zda je fronta plná.

Publikování/odběr

K porovnání publikování s dostupnými odběry dojde, když aplikace JMS odešle zprávu s nenulovým zpožděním doručení. Zpráva pro každého odpovídajícího odběratele je vložena do SYSTEM.DDELAY.LOCAL.QUEUE , kde se uchovává, dokud se nedokončí prodleva doručení. Je-li jedním z těchto odběratelů proxy odběr pro jiného správce front, dojde po dokončení prodlevy doručení k větvení tohoto správce front. To může vést k tomu, že odběratelé v druhém správci front obdrží publikování, která byla původně publikována před přihlášením k odběru. Jedná se o odchylku od specifikace JMS 2.0 nebo novější.

Prodleva doručení s publikováním/odběrem je podporována pouze v případě, že je cílové téma nakonfigurováno s (N) PMSGDLV = ALLAVAIL. Při pokusu o použití jiných hodnot dojde k chybě MQRC_PUBLICATION_FAILURE. Pokud při vkládání zprávy do cílové fronty dojde k selhání procesoru prodlevy doručení, bude výsledek stejný, jako je popsáno v části "Selhání při vkládání do cílové fronty".

Zprávy sestavy

Všechny volby sestavy jsou podporovány a podporovány procesorem doručení, jiné než následující volby, které jsou ignorovány, ale předány ve zprávě, když je odeslána do cílové fronty:

- MQRO_COA*
- MQRO_COD*
- MQRO_PAN/MQRO_NAN
- 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 používají klonované odběry nebo sdílené odběry.

Klonované odběry

Klonovaný odběr je rozšíření IBM MQ . Klonované odběry umožňují více odběratelům v různých prostředích JVM (Java Virtual Machine) souběžný přístup k odběru. Toto chování lze použít nastavením vlastnosti **CLONESUPP** na hodnotu Povoleno v objektu ConnectionFactory . Standardně **CLONESUPP** je Zakázáno. Klonované odběry lze povolit pouze u trvalých odběrů. Je-li povolena volba **CLONESUPP** , jsou klonována všechna následná připojení, která jsou vytvořena pomocí této továrny ConnectionFactory .

Trvalý odběr lze považovat za klonovaný, pokud je vytvořen jeden nebo více spotřebitelů pro příjem zpráv z tohoto odběru, tj. byl vytvořen se stejným názvem odběru. To lze provést pouze v případě, že připojení, v rámci kterého byli spotřebitelé vytvořeni, je **CLONESUPP** nastaveno na hodnotu `Povoleno` na MQConnectionFactory. Je-li v tématu odběru publikována zpráva, odešle se do odběru kopie této zprávy. Zpráva je k dispozici všem spotřebitelům, ale pouze jeden ji obdrží.

Poznámka: Povolení klonovaných odběrů rozšiřuje specifikaci JMS .

Sdílené odběry

Sdílené odběry, které byly zavedeny specifikací JMS 2.0 , umožňují sdílení zpráv z odběru tématu mezi více spotřebiteli. Každá zpráva z odběru je doručena pouze jednomu ze spotřebitelů daného odběru. Sdílené odběry jsou povoleny příslušným voláním rozhraní API JMS 2.0 nebo novější.

Rozhraní API lze volat jedním z následujících způsobů:

- Z aplikace Java SE (nebo kontejneru klienta Java EE).
- Ze servletu nebo implementace MDB.

Specifikace JMS 2.0 nebo novější nedefinuje žádný standardní způsob řízení objektu MDB ze sdíleného odběru, takže produkt IBM MQ 8.0 nebo novější poskytuje pro tento účel vlastnost specifikace aktivace **`sharedSubscription`** . Další informace o této vlastnosti viz “Konfigurace adaptéru prostředků pro příchozí komunikaci” na stránce 439 a “Příklady, jak definovat vlastnost sharedSubscription” na stránce 456.

Je-li povolen sdílený odběr, nelze zrušit jeho sdílení.

Sdílené odběry lze vytvářet jako trvalé nebo dočasné odběry. Není nutné samostatně vytvářet žádné objekty na straně správce front nad rámec normální konfigurace produktu JMS . Všechny požadované objekty jsou vytvářeny dynamicky.

Rozhodování mezi sdílenými nebo klonovanými odběry

Při rozhodování, zda použít sdílené nebo klonované odběry, zvažte výhody obou. Je-li to možné, používejte sdílené odběry, protože se jedná o chování definované specifikací, a nikoli o specifické rozšíření IBM MQ .

Následující tabulka obsahuje některé body, které je třeba zvážit při rozhodování mezi sdílenými a klonovanými odběry:

Tabulka 48. Porovnání aspektů pro sdílené odběry a klonované odběry

Sdílené odběry	**Klonované odběry**
Sdílené odběry jsou standardní součástí specifikace JMS 2.0 nebo novější.	Klonované odběry jsou specifické rozšíření IBM MQ .
Sdílené odběry jsou vytvářeny pomocí explicitních volání metody rozhraní API.	Klonované odběry jsou řízeny administrativně na úrovni ConnectionFactory .
Sdílené odběry mohou být trvalé nebo přechodné.	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.
Pokud je odběr vytvořen jako sdílený, nelze jej později změnit na nesdílený a naopak.	Odběr lze změnit z klonovaného na neklonovaný při každém opětovném otevření, pokud se změnila vlastnost **CLONESUPP** vlastnícího připojení.

Související pojmy

Odběratelé a odběry

Trvanlivost předplatného

Související úlohy

Použití sdílených odběrů JMS 2.0

Související odkazy

“Příklady, jak definovat vlastnost sharedSubscription” na stránce 456

Můžete definovat vlastnost sharedSubscription specifikace aktivace v rámci souboru WebSphere Liberty `server.xml` . Případně můžete definovat vlastnost v rámci objektu typu message-driven bean (MDB) pomocí anotací.

CLONESUPP

Vlastnost SupportMQExtensions

Specifikace JMS 2.0 zavedla změny způsobu práce s určitým chováním. Produkt IBM MQ 8.0 a novější obsahuje vlastnost **`com.ibm.mq.jms.SupportMQExtensions`**, kterou lze nastavit na hodnotu TRUE , aby se tyto změněné chování vrátily zpět na předchozí implementace.

V 9.3.0 JM 3.0 V 9.3.0 Vlastnost **`com.ibm.mq.jakarta.jms.SupportMQExtensions`** (Jakarta Messaging 3.0) je podporována produktem IBM MQ classes for Jakarta Messaging, který je k dispozici v produktu `com.ibm.mq.jakarta.client.jar`.

JMS 2.0 Vlastnost **`com.ibm.mq.jms.SupportMQExtensions`** (JMS 2.0) je podporována produktem IBM MQ classes for JMS, který je k dispozici v adresáři `com.ibm.mq.allclient.jar` nebo `com.ibm.mqjms.jar`.

Tři oblasti funkčnosti jsou vráceny zpět nastavením **`SupportMQExtensions`** na hodnotu `True`:

Priorita zprávy
: Zprávám lze přiřadit prioritu 0-9. Před produktem JMS 2.0mohou zprávy také používat hodnotu -1, což znamená, že je použita výchozí priorita fronty. JMS 2.0 a novější nepovolují nastavení priority zprávy -1. Zapnutí **`SupportMQExtensions`** umožňuje použít hodnotu -1.

ID klienta
: Specifikace JMS 2.0 nebo novější vyžaduje, aby byla při vytváření připojení kontrolována jedinečnost nenulového ID klienta. Zapnutí parametru **`SupportMQExtensions`** znamená, že tento požadavek nebude zohledněn a že ID klienta lze znovu použít.

NoLocal
: Specifikace JMS 2.0 nebo novější vyžaduje, aby při zapnutí této konstanty spotřebitel nemohl přijímat zprávy publikované stejným ID klienta. Před JMS 2.0byl tento atribut nastaven na odběrateli, aby zabránil přijímání zpráv, které jsou publikovány jeho vlastním připojením. Zapnutí produktu **`SupportMQExtensions`** vrátí toto chování zpět na předchozí implementaci.

Tuto vlastnost lze nastavit takto:

```
java -Dcom.ibm.mq.jms.SupportMQExtensions=true
```

Tuto vlastnost lze nastavit buď jako standardní systémovou vlastnost prostředí JVM v příkazu **`java`** , nebo obsaženou v konfiguračním souboru IBM MQ classes for JMS .

Související pojmy

“Konfigurační soubor IBM MQ classes for JMS/Jakarta Messaging” na stránce 95

Konfigurační soubory IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging uvádějí vlastnosti, které se používají ke konfiguraci IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging.

Související odkazy

“Vlastnosti použité ke konfiguraci chování klienta JMS” na stránce 101

Pomocí těchto vlastností můžete konfigurovat chování klienta JMS .

Použití sdílených odběrů v aplikacích JMS

V případě sdílených odběrů je jeden odběr sdílen mezi více spotřebiteli, přičemž v libovolném okamžiku obdrží publikaci pouze jeden ze spotřebitelů.

Sdílené odběry jsou k dispozici od JMS 2.0 . Při vývoji aplikace JMS pro produkt IBM MQ 8.0 nebo novější proto může být nutné zvážit dopad této funkce na správce front.

Myšlenkou sdílených odběrů je sdílení zátěže mezi více spotřebiteli. Trvalý odběr lze také sdílet mezi více spotřebiteli.

Předpokládejme například, že existuje:

- Odběr SUB, přihlášení k odběru tématu FIFA2014/UPDATES pro příjem aktualizací fotbalových zápasů, sdílení třemi spotřebiteli C1, C2a C3 .
- Producent P1 publikující v tématu FIFA2014/UPDATES

Po provedení publikace FIFA2014/UPDATESobdrží publikaci pouze jeden ze tří odběratelů (C1, C2nebo C3), nikoli však všichni.

Následující ukázka demonstruje použití sdílených odběrů a také demonstruje použití dalšího rozhraní API v produktu JMS 2.0, `Message.receiveBody()` k načtení pouze těla zprávy.

Ukázka vytvoří tři podprocesy odběratele, které vytvoří sdílený odběr tématu FIFA2014/UPDATES a jeden podproces vydavatele.

V 9.3.0 JM 3.0 V 9.3.0

```
package mqv91Samples;

import jakarta.jms.JMSException;

import com.ibm.msg.client.jms.JmsConnectionFactory;
import com.ibm.msg.client.jms.JmsFactoryFactory;
import com.ibm.msg.client.wmq.WMQConstants;

import jakarta.jms.JMSContext;
import jakarta.jms.Topic;
import jakarta.jms.Queue;
import jakarta.jms.JMSConsumer;
import jakarta.jms.Message;
import jakarta.jms.JMSProducer;

/*
* Implements both Subscriber and Publisher
*/
class SharedNonDurableSubscriberAndPublisher implements Runnable {
   private Thread t;
   private String threadName;

   SharedNonDurableSubscriberAndPublisher( String name){
       threadName = name;
       System.out.println("Creating Thread:" +  threadName );
   }

   /*
    * Demonstrates shared non-durable subscription in JMS 2.0 and later
    */
    private void sharedNonDurableSubscriptionDemo(){
        JmsConnectionFactory cf = null;
        JMSContext msgContext = null;

        try {
            // Create Factory for WMQ JMS provider
            JmsFactoryFactory ff = JmsFactoryFactory.getInstance(WMQConstants.WMQ_PROVIDER);
            // Create connection factory
            cf = ff.createConnectionFactory();
            // Set MQ properties
            cf.setStringProperty(WMQConstants.WMQ_QUEUE_MANAGER, "QM3");
            cf.setIntProperty(WMQConstants.WMQ_CONNECTION_MODE, WMQConstants.WMQ_CM_BINDINGS);
            // Create message context
            msgContext = cf.createContext();

            // Create a topic destination
            Topic fifaScores = msgContext.createTopic("/FIFA2014/UPDATES");

            // Create a consumer. Subscription name specified, required for sharing of subscription.
            JMSConsumer msgCons = msgContext.createSharedConsumer(fifaScores, "FIFA2014SUBID");

            // Loop around to receive publications
            while(true){
```

```
                String msgBody=null;

                // Use JMS 2.0 and later receiveBody method as we are interested in message body only.
                msgBody = msgCons.receiveBody(String.class);

                if(msgBody != null){
                    System.out.println(threadName + " : " + msgBody);
                }
            }
        }catch(JMSException jmsEx){
            System.out.println(jmsEx);
        }
    }
```

JMS 2.0

```
package mqv91Samples;

import javax.jms.JMSException;

import com.ibm.msg.client.jms.JmsConnectionFactory;
import com.ibm.msg.client.jms.JmsFactoryFactory;
import com.ibm.msg.client.wmq.WMQConstants;

import javax.jms.JMSContext;
import javax.jms.Topic;
import javax.jms.Queue;
import javax.jms.JMSConsumer;
import javax.jms.Message;
import javax.jms.JMSProducer;

/*
* Implements both Subscriber and Publisher
*/
class SharedNonDurableSubscriberAndPublisher implements Runnable {
   private Thread t;
   private String threadName;

   SharedNonDurableSubscriberAndPublisher( String name){
       threadName = name;
       System.out.println("Creating Thread:" +  threadName );
   }

   /*
    * Demonstrates shared non-durable subscription in JMS 2.0 and later
    */
    private void sharedNonDurableSubscriptionDemo(){
        JmsConnectionFactory cf = null;
        JMSContext msgContext = null;

        try {
            // Create Factory for WMQ JMS provider
            JmsFactoryFactory ff = JmsFactoryFactory.getInstance(WMQConstants.WMQ_PROVIDER);
            // Create connection factory
            cf = ff.createConnectionFactory();
            // Set MQ properties
            cf.setStringProperty(WMQConstants.WMQ_QUEUE_MANAGER, "QM3");
            cf.setIntProperty(WMQConstants.WMQ_CONNECTION_MODE, WMQConstants.WMQ_CM_BINDINGS);
            // Create message context
            msgContext = cf.createContext();

            // Create a topic destination
            Topic fifaScores = msgContext.createTopic("/FIFA2014/UPDATES");

            // Create a consumer. Subscription name specified, required for sharing of subscription.
            JMSConsumer msgCons = msgContext.createSharedConsumer(fifaScores, "FIFA2014SUBID");

            // Loop around to receive publications
            while(true){

                String msgBody=null;

                // Use JMS 2.0 and later receiveBody method as we are interested in message body only.
                msgBody = msgCons.receiveBody(String.class);

                if(msgBody != null){
                    System.out.println(threadName + " : " + msgBody);
                }
            }
        }catch(JMSException jmsEx){
            System.out.println(jmsEx);
```

```
        }
    }
```

```
 /*
        * Publisher publishes match updates like current attendance in the stadium, goal score and ball
possession by teams.
        */
        private void matchUpdatePublisher(){
            JmsConnectionFactory cf = null;
            JMSContext msgContext = null;
            int nederlandsGoals = 0;
            int chileGoals = 0;
            int stadiumAttendence = 23231;
            int switchIndex = 0;
            String msgBody = "";
            int nederlandsHolding = 60;
            int chileHolding = 40;

            try {
                // Create Factory for WMQ JMS provider
                JmsFactoryFactory ff = JmsFactoryFactory.getInstance(WMQConstants.WMQ_PROVIDER);

                // Create connection factory
                cf = ff.createConnectionFactory();
                // Set MQ properties
                cf.setStringProperty(WMQConstants.WMQ_QUEUE_MANAGER, "QM3");
                cf.setIntProperty(WMQConstants.WMQ_CONNECTION_MODE, WMQConstants.WMQ_CM_BINDINGS);

                // Create message context
                msgContext = cf.createContext();

                // Create a topic destination
                Topic fifaScores = msgContext.createTopic("/FIFA2014/UPDATES");

                // Create publisher to publish updates from stadium
                JMSProducer msgProducer = msgContext.createProducer();

                while(true){
                    // Send match updates
                    switch(switchIndex){
                        // Attendance
                    case 0:
                        msgBody ="Stadium Attendence " + stadiumAttendence;
                        stadiumAttendence += 314;
                        break;

                        // Goals
                    case 1:
                        msgBody ="SCORE: The Netherlands: " + nederlandsGoals + " - Chile:" + chileGoals;
                        break;

                        // Ball possession percentage
                    case 2:
                        msgBody ="Ball possession: The Netherlands: " + nederlandsHolding + "% - Chile:
" + chileHolding + "%";
                        if((nederlandsHolding > 60) && (nederlandsHolding < 70)){
                            nederlandsHolding -= 2;
                            chileHolding += 2;
                        }else{
                            nederlandsHolding += 2;
                            chileHolding -= 2;
                        }
                        break;
                    }

                    // Publish and wait for two seconds to publish next update
                    msgProducer.send (fifaScores, msgBody);
                    try{
                        Thread.sleep(2000);
                    }catch(InterruptedException iex){

                    }

                    // Increment and reset the index if greater than 2
                    switchIndex++;
                    if(switchIndex > 2)
                        switchIndex = 0;
                }
            }catch(JMSException jmsEx){
                System.out.println(jmsEx);
            }
```

```
        }

    /*
     * (non-Javadoc)
     * @see java.lang.Runnable#run()
     */
    public void run() {
        // If this is a publisher thread
        if(threadName == "PUBLISHER"){
            matchUpdatePublisher();
        }else{
            // Create subscription and start receiving publications
            sharedNonDurableSubscriptionDemo();
        }
    }

       // Start thread
   public void start (){
      System.out.println("Starting " +  threadName );
      if (t == null)
      {
         t = new Thread (this, threadName);
         t.start ();
      }
   }
}
```

```
/*
* Demonstrate JMS 2.0 and later simplified API using IBM MQ 91 JMS Implementation
*/
public class Mqv91jms2Sample {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        // Create first subscriber and start
        SharedNonDurableSubscriberAndPublisher subOne = new
SharedNonDurableSubscriberAndPublisher( "SUB1");
        subOne.start();

        // Create second subscriber and start
        SharedNonDurableSubscriberAndPublisher subTwo = new
SharedNonDurableSubscriberAndPublisher( "SUB2");
        subTwo.start();

        // Create third subscriber and start
        SharedNonDurableSubscriberAndPublisher subThree = new
SharedNonDurableSubscriberAndPublisher( "SUB3");
        subThree.start();

        // Create publisher and start
        SharedNonDurableSubscriberAndPublisher publisher = new
SharedNonDurableSubscriberAndPublisher( "PUBLISHER");
        publisher.start();
    }
}
```

Související pojmy

Jazyková rozhraní produktu IBM MQ Java

V 9.3.2 Konfigurace modulární aplikace pro použití IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging

V 9.3.2 Moduly IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging můžete používat modulárním způsobem, a to tak, že budete vyžadovat příslušný modul ve své aplikaci a zahrnete příslušný adresář do cesty k modulu.

Modulární balení

Unifikované soubory JAR pro soubory IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging poskytují automatické názvy modulů, které nahrazují výchozí názvy odvozené z názvů souborů JAR.

- IBM MQ classes for JMS (`com.ibm.mq.allclient.jar`) jsou poskytovány s názvem modulu `com.ibm.mq.javax`.
- IBM MQ classes for Jakarta Messaging (`com.ibm.mq.jakarta.client.jar`) jsou poskytovány s názvem modulu `com.ibm.mq.jakarta`.

Výchozí adresář `MQ_HOME/java/lib` není vhodný pro modulární použití, protože moduly nemohou obsahovat stejný balík a výchozí adresář obsahuje stejné balíky ve více adresářích JAR. Proto jsou k dispozici nové adresáře, které obsahují pouze potřebné soubory JAR bez duplikace balíků mezi soubory JAR. Tyto adresáře jsou vhodné pro zahrnutí na `module-path`.

Poznámka: Pokud máte aplikace, které používají dostupné soubory JAR v modulárním kontextu a spoléhají se na výchozí názvy modulů, musíte aktualizovat své aplikace tak, aby vyžadovaly nové názvy modulů. Výchozí názvy modulů jsou odvozeny z názvů souborů JAR.

Konfigurace modulární aplikace pro použití IBM MQ classes for JMS

Modulární aplikaci můžete nakonfigurovat tak, aby používala produkt IBM MQ classes for JMS (`com.ibm.mq.allclient.jar`), a to takto:

- Nakonfigurujte aplikaci tak, aby vyžadovala modul `com.ibm.mq.javax` .
- Nakonfigurujte aplikaci tak, aby obsahovala adresář `MQ_HOME/java/lib/modules/javax` v cestě k modulu.

Konfigurace modulární aplikace pro použití IBM MQ classes for Jakarta Messaging

Modulární aplikaci můžete nakonfigurovat tak, aby používala produkt IBM MQ classes for Jakarta Messaging (`com.ibm.mq.jakarta.client.jar`), a to takto:

- Nakonfigurujte aplikaci tak, aby vyžadovala modul `com.ibm.mq.jakarta` .
- Nakonfigurujte aplikaci tak, aby obsahovala adresář `MQ_HOME/java/lib/modules/jakarta` v cestě k modulu.

Konfigurace modulární aplikace pro použití IBM MQ classes for Java

Chcete-li použít IBM MQ classes for Java z modulární aplikace, můžete použít buď konfiguraci pro IBM MQ classes for JMS , nebo konfiguraci pro IBM MQ classes for Jakarta Messaging, protože oba soubory JAR klienta podporují IBM MQ classes for Java. Vaše aplikace však musí používat pouze jednu z těchto konfigurací, nikoli obě.

IBM MQ classes for JMS Zařízení aplikačního serveru

Toto téma popisuje, jak produkt IBM MQ classes for JMS implementuje třídu ConnectionConsumer a rozšířené funkce ve třídě Session. Také shrnuje funkci fondu relací serveru.

Důležité: Tyto informace jsou pouze orientační. Aplikace nesmí být zapsána pro použití tohoto rozhraní: používá se v rámci adaptéru prostředků IBM MQ pro připojení k serverům Java EE. Praktické informace o připojení viz “Použití adaptéru prostředků IBM MQ” na stránce 423.

Produkt IBM MQ classes for JMS podporuje zařízení ASF (Application Server Facilities), která jsou uvedena v *Java Message Service specifikaci* (viz Oracle Technology Network for Java Developers). Tato specifikace identifikuje tři role v rámci tohoto programovacího modelu:

- **Poskytovatel JMS provider** poskytuje ConnectionConsumer a rozšířené funkce relace.
- **Aplikační server** poskytuje funkce ServerSessionPool a ServerSession .
- **Aplikace klienta** používá funkčnost, kterou poskytuje poskytovatel JMS a aplikační server.

Informace v tomto tématu se nepoužijí, pokud aplikace používá připojení v reálném čase ke zprostředkovateli.

JMS ConnectionConsumer

Rozhraní ConnectionConsumer poskytuje vysoce výkonnou metodu souběžného doručování zpráv do fondu podprocesů.

Specifikace JMS umožňuje aplikačnímu serveru úzkou integraci s implementací produktu JMS pomocí rozhraní `ConnectionConsumer` . Tato funkce poskytuje souběžné zpracování zpráv. Aplikační server obvykle vytvoří fond podprocesů a implementace JMS zpřístupní těmto podprocesům zprávy. Aplikační server se systémem JMS(například WebSphere Application Server) může tuto funkci používat k poskytování funkcí systému zpráv vysoké úrovně, například objektů typu message-driven bean.

Běžné aplikace nepoužívají objekt ConnectionConsumer, ale zkušení klienti JMS jej mohou používat. Pro takové klienty ConnectionConsumer poskytuje vysoce výkonnou metodu pro souběžné doručování zpráv do fondu podprocesů. Když zpráva dorazí do fronty nebo tématu, produkt JMS vybere podproces z fondu a doručí mu dávku zpráv. Chcete-li to provést, JMS spustí přidruženou metodu MessageListener' s `onMessage()` .

Stejného efektu lze dosáhnout vytvořením více objektů Session a MessageConsumer , z nichž každý má registrovaný modul MessageListener. Hodnota ConnectionConsumer však poskytuje lepší výkon, menší využití prostředků a větší flexibilitu. Zejména je vyžadováno méně objektů relace.

Plánování aplikace pomocí funkce ASF

Tento oddíl vám řekne, jak naplánovat aplikaci, včetně:

- “Obecné zásady pro dvoubodové zasílání zpráv pomocí ASF” na stránce 323
- “Obecné zásady pro publikování/odběr zpráv pomocí ASF” na stránce 324
- “Odebrání zpráv z fronty v ASF” na stránce 325
- Zpracování nezpracovatelných zpráv v ASF. Viz “Zpracování nezpracovatelných zpráv v adresáři IBM MQ classes for JMS” na stránce 225.

Obecné zásady pro dvoubodové zasílání zpráv pomocí ASF

Toto téma obsahuje obecné informace o systému zpráv typu point-to-point s použitím funkce ASF.

Když aplikace vytvoří objekt ConnectionConsumer z objektu QueueConnection , určuje objekt fronty JMS a řetězec selektoru. Objekt ConnectionConsumer poté začne poskytovat zprávy relacím v přidruženém fondu ServerSession. Zprávy přicházejí do fronty a pokud se shodují s selektorem, jsou doručeny do relací v přidruženém fondu ServerSession.

V termínech IBM MQ odkazuje objekt fronty na QLOCAL nebo QALIAS v lokálním správci front. Pokud se jedná o QALIAS, musí tento QALIAS odkazovat na QLOCAL. Plně vyřešený IBM MQ QLOCAL je znám jako *základní QLOCAL*. Hodnota ConnectionConsumer je *aktivní* , pokud není zavřená a její nadřízený objekt QueueConnection je spuštěn.

Je možné, aby více ConnectionConsumers, každý s různými selektory, bylo spuštěno na stejném základním QLOCAL. Chcete-li zachovat výkon, nesmí se ve frontě hromadit nežádoucí zprávy. Nežádoucí zprávy jsou ty, pro které žádný aktivní objekt ConnectionConsumer nemá odpovídající selektor. Továrnu QueueConnectionmůžete nastavit tak, aby byly tyto nežádoucí zprávy z fronty odebrány (podrobnosti viz “Odebrání zpráv z fronty v ASF” na stránce 325). Toto chování můžete nastavit jedním ze dvou způsobů:

- Pomocí nástroje pro administraci JMS nastavte továrnu QueueConnectionna hodnotu MRET (NO).
- Ve svém programu použijte:

```
MQQueueConnectionFactory.setMessageRetention(WMQConstants.WMQ_MRET_NO)
```

Pokud toto nastavení nezměníte, standardně se tyto nežádoucí zprávy ve frontě uchovávají.

Při nastavování správce front IBM MQ zvažte 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 povolenou frontu nedoručených zpráv. Pokud se při vkládání zprávy do fronty nedoručených zpráv vyskytne problém s konzumentem připojení ConnectionConsumer , zastaví se doručování zpráv z podkladového QLOCAL. Chcete-li definovat frontu nedoručených zpráv, použijte:

```
ALTER QMGR DEADQ( your.dead.letter.queue.name )
```

- Uživatel, který spouští ConnectionConsumer , musí mít oprávnění k provedení MQOPEN s MQOOO_SAVE_ALL_CONTEXT a MQOOO_PASS_ALL_CONTEXT. Podrobnosti naleznete v dokumentaci k produktu IBM MQ pro vaši specifickou platformu.
- Pokud jsou nežádoucí zprávy ponechány ve frontě, sníží výkon systému. Proto naplánujte selektory zpráv tak, aby mezi nimi uživatelé ConnectionConsumers odebrali všechny zprávy z fronty.

Podrobnosti o příkazech MQSC viz Příkazy MQSC.

Obecné zásady pro publikování/odběr zpráv pomocí ASF
Volba ConnectionConsumers přijímá zprávy pro určené téma. Hodnota ConnectionConsumer může být trvalá nebo netrvalá. Musíte určit, kterou frontu nebo fronty používá ConnectionConsumer .

Když aplikace vytvoří objekt ConnectionConsumer z objektu TopicConnection , určí objekt Topic a řetězec selektoru. Objekt ConnectionConsumer poté začne přijímat zprávy, které odpovídají selektoru daného tématu, včetně všech zachovaných publikování pro odebírané téma.

Alternativně může aplikace vytvořit trvalý objekt ConnectionConsumer , který je přidružen ke specifickému názvu. Tento ConnectionConsumer přijímá zprávy, které byly publikovány v tématu od doby, kdy byl trvalý ConnectionConsumer naposledy aktivní. Obdrží všechny takové zprávy, které odpovídají selektoru v tématu. Pokud však ConnectionConsumer používá dopředné čtení, může při zavření ztratit přechodné zprávy, které jsou ve vyrovnávací paměti klienta.

Pokud se produkt IBM MQ classes for JMS nachází v režimu migrace poskytovatele systému zpráv IBM MQ , je pro dočasné odběry ConnectionConsumer použita samostatná fronta.Konfigurovatelná volba CCSUB pro továrnu TopicConnectionurčuje frontu, která má být použita. Obvykle CCSID určuje jednu frontu pro použití všemi ConnectionConsumers , kteří používají stejnou továrnu TopicConnection. Je však možné, aby každý ConnectionConsumer generoval dočasnou frontu zadáním předpony názvu fronty následované hvězdičkou (*).

Pokud se produkt IBM MQ classes for JMS nachází v režimu migrace poskytovatele systému zpráv IBM MQ , vlastnost CCDSUB tématu určuje frontu, která má být použita pro trvalé odběry. Opět to může být fronta, která již existuje, nebo předpona názvu fronty následovaná hvězdičkou (*). Zadáte-li frontu, která již existuje, budou tuto frontu používat všichni trvalí uživatelé ConnectionConsumers , kteří se přihlásí k odběru tématu. Zadáte-li předponu názvu fronty následovanou hvězdičkou (*), bude při prvním vytvoření trvalého objektu ConnectionConsumer s konkrétním názvem vygenerována fronta. Tato fronta je znovu použita později při vytvoření trvalého objektu ConnectionConsumer se stejným názvem.

Při nastavování správce front IBM MQ zvažte následující body:

- Váš správce front musí mít povolenou frontu nedoručených zpráv. Pokud se při vkládání zprávy do fronty nedoručených zpráv vyskytne problém s konzumentem připojení ConnectionConsumer , zastaví se doručování zpráv z podkladového QLOCAL. Chcete-li definovat frontu nedoručených zpráv, použijte:

```
ALTER QMGR DEADQ( your.dead.letter.queue.name )
```

- Uživatel, který spouští ConnectionConsumer , musí mít oprávnění k provedení MQOPEN s MQOOO_SAVE_ALL_CONTEXT a MQOOO_PASS_ALL_CONTEXT. Podrobnosti naleznete v dokumentaci k produktu IBM MQ pro vaši platformu.
- Výkon pro jednotlivé ConnectionConsumer můžete optimalizovat vytvořením samostatné vyhrazené fronty. Jedná se o náklady na další využití prostředků.

Odebrání zpráv z fronty v ASF
Pokud aplikace používá volbu ConnectionConsumers, může být nutné, aby produkt JMS v řadě situací odebral zprávy z fronty.

Tyto situace jsou následující:

Nesprávně formátovaná zpráva
Může být doručena zpráva, kterou produkt JMS nemůže analyzovat.

Nezpracovatelná zpráva
Zpráva může dosáhnout prahové hodnoty vrácení, ale ConnectionConsumer ji nedokáže znovu zařadit do fronty vrácení.

Není zájem o ConnectionConsumer
Pro systém zpráv typu point-to-point platí, že pokud je továrna QueueConnectionnastavena tak, aby neuchovávala nežádoucí zprávy, přijde zpráva, která je nechtěná pro některého z ConnectionConsumers.

V těchto situacích se ConnectionConsumer pokusí odebrat zprávu z fronty. Volby odebrání v poli sestavy deskriptoru 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 je vyřazena.

ConnectionConsumer také generuje zprávu sestavy, což také závisí na poli sestavy deskriptoru MQMD zprávy. Tato zpráva je odeslána do fronty ReplyTozprávy na správci front ReplyTo. Pokud se při odesílání zprávy sestavy vyskytne chyba, zpráva se místo toho odešle do fronty nedoručených zpráv. Volby sestavy výjimek v poli sestavy podrobností sady MQMD zprávy sestavy. Tyto volby jsou:

MQRO_EXCEPTION
Vygeneruje se zpráva sestavy, která obsahuje MQMD původní zprávy. Neobsahuje žádná data těla zprávy.

MQRO_EXCEPTION_WITH_DATA
Vygeneruje se zpráva sestavy, která obsahuje MQMD, libovolná záhlaví MQ a 100 bajtů dat těla.

MQRO_EXCEPTION_WITH_FULL_DATA
Vygeneruje se zpráva sestavy, která obsahuje všechna data z původní zprávy.

default
Není generována žádná zpráva sestavy.

Při generování zpráv sestavy jsou uznány následující volby:

- MQRO_NEW_MSG_ID
- MQRO_PASS_MSG_ID
- MQRO_COPY_MSG_ID_TO_CORREL_ID
- MQRO_PASS_CORREL_ID

Pokud nezpracovatelnou zprávu nelze znovu zařadit do fronty, například proto, že fronta nedoručených zpráv je plná nebo je nesprávně zadána autorizace, závisí to, co se stane, na perzistenci zprávy. Je-li zpráva dočasná, je vyřazena a není generována žádná zpráva sestavy. Pokud je zpráva trvalá, doručení zpráv všem spotřebitelům připojení, kteří naslouchají na daném místě určení, se zastaví. Takoví spotřebitelé připojení musí být zavřeni a problém musí být vyřešen před opětovným vytvořením a restartováním doručení zprávy.

Je důležité definovat frontu nedoručených zpráv a pravidelně ji kontrolovat, aby nedošlo k žádným problémům. Zejména se ujistěte, že fronta nedoručených zpráv nedosahuje své maximální hloubky a že je její maximální velikost zpráv dostatečně velká pro všechny zprávy.

Když je zpráva znovu zařazena do fronty nedoručených zpráv, předchází jí IBM MQ záhlaví nedoručených zpráv (MQDLH). Podrobné informace o formátu MQDLH naleznete v tématu Záhlaví nedoručených zpráv

MQDLH . Můžete identifikovat zprávy, které ConnectionConsumer umístil do fronty nedoručených zpráv, nebo zprávy, které ConnectionConsumer vygeneroval, pomocí následujících polí:

- PutApplTyp je `MQAT_JAVA (0x1C)`
- PutApplNázev je "`MQ JMS ConnectionConsumer`"

Tato pole jsou v MQDLH zpráv ve frontě nedoručených zpráv a v MQMD zpráv sestav. Pole zpětné vazby MQMD a pole Příčina MQDLH obsahují kód popisující chybu. Podrobnosti o těchto kódech viz "Kódy příčiny a zpětné vazby v ASF" na stránce 327. Další pole jsou popsána v části MQDLH-záhlaví nedoručených zpráv.

Zpracování nezpracovatelných zpráv v ASF
V rámci zařízení aplikačního serveru je zpracování nezpracovatelných zpráv ošetřeno mírně jinak než jinde v produktu IBM MQ classes for JMS.

Informace o zpracování nezpracovatelných zpráv v souboru IBM MQ classes for JMSviz "Zpracování nezpracovatelných zpráv v adresáři IBM MQ classes for JMS" na stránce 225.

Používáte-li funkce ASF (Application Server Facilities), ConnectionConsumernamísto MessageConsumerzpracovává nezpracovatelné zprávy. ConnectionConsumer požaduje zprávy podle vlastností fronty BackoutThreshold a BackoutRequeue.

Když aplikace používá ConnectionConsumers, závisí okolnosti, za kterých je zpráva vrácena, na relaci, kterou poskytuje aplikační server:

- Je-li relace netransakční s parametrem AUTO_KVITOVAT nebo DUPS_OK_KVITOVAT, je zpráva vrácena zpět pouze po chybě systému nebo v případě neočekávaného ukončení aplikace.
- Je-li relace netransakční s volbou CLIENT_ACKNOWLEDGED, mohou být nepotvrzené zprávy zálohovány aplikačním serverem, který volá metodu Session.recover().

  Implementace klienta MessageListener nebo aplikační server obvykle volá funkci Message.acknowledge(). Message.acknowledge() potvrzuje všechny zprávy dosud doručené v relaci.
- Je-li relace transakční, mohou být nepotvrzené zprávy zálohovány aplikačním serverem, který volá funkci Session.rollback().
- Pokud aplikační server dodává parametr XASession, zprávy jsou potvrzeny nebo vráceny zpět v závislosti na distribuované transakci. Aplikační server přebírá odpovědnost za dokončení transakce.

Související pojmy
"Zpracování nezpracovatelných zpráv v adresáři IBM MQ classes for JMS" na stránce 225
Nezpracovatelnou zprávou je zpráva, kterou přijímající aplikace nemůže zpracovat. Je-li aplikace doručena nezpracovatelnou zprávu a byla-li několikrát odvolána, produkt IBM MQ classes for JMS ji může přesunout do fronty vrácení.

Ošetření chyb

Tento oddíl se zabývá různými aspekty ošetření chyb, včetně "Obnova z chybových stavů v ASF" na stránce 326 a "Kódy příčiny a zpětné vazby v ASF" na stránce 327.

Obnova z chybových stavů v ASF
Pokud dojde k závažné chybě ConnectionConsumer , zastaví se doručení zpráv všem ConnectionConsumers se zájmem o stejný QLOCAL. Dojde-li k této situaci, bude upozorněn modul ExceptionListener , který je registrován u ovlivněného připojení. Existují dva způsoby, jak se může aplikace zotavit z těchto chybových stavů.

K závažné chybě tohoto druhu obvykle dochází v případě, že objekt ConnectionConsumer nemůže zprávu znovu zařadit do fronty nedoručených zpráv nebo dojde k chybě při čtení zpráv z QLOCAL.

Vzhledem k tomu, že je upozorněn modul ExceptionListener , který je registrován s ovlivněným připojením, můžete jej použít k identifikaci příčiny problému. V některých případech musí administrátor systému zasáhnout, aby problém vyřešil.

Chcete-li se zotavit z těchto chybových stavů, použijte jednu z následujících technik:

- Zavolejte `close()` na všechny ovlivněné ConnectionConsumers. Aplikace může vytvořit nové ConnectionConsumers pouze po zavření všech ovlivněných ConnectionConsumers a vyřešení všech systémových problémů.
- Zavolejte `stop()` na všechna ovlivněná připojení. Po zastavení všech připojení a vyřešení případných problémů se systémem může aplikace `start()` úspěšně provést svá připojení.

Kódy příčiny a zpětné vazby v ASF
K určení příčiny chyby použijte kódy příčiny a zpětné vazby. Zde jsou uvedeny obecné kódy příčiny generované konzumentem připojení ConnectionConsumer .

Chcete-li určit příčinu chyby, použijte následující informace:

- Kód zpětné vazby v jakýchkoli zprávách sestavy
- Kód příčiny v MQDLH všech zpráv ve frontě nedoručených zpráv

ConnectionConsumers generují následující kódy příčiny.

MQRC_BACKOUT_THRESHOLD_DOSAŽENÍ (0x93A; 2362)

Příčina
Zpráva dosáhla prahové hodnoty vrácení definované na QLOCAL, ale není definována žádná fronta vrácení.

Na platformách, kde nelze definovat frontu vrácení, dosáhla zpráva prahové hodnoty JMS-defined backout 20.

Akce
Pokud to není žádoucí, definujte frontu vrácení pro odpovídající QLOCAL. Podívejte se také na příčinu vícenásobných vrácení.

MQRC_MSG_NOT_MATCHED (0x93B; 2363)

Příčina
V systému zpráv typu point-to-point existuje zpráva, která neodpovídá žádnému ze selektorů pro monitorování fronty ConnectionConsumers . Pro zachování výkonu je zpráva znovu zařazena do fronty nedoručených zpráv.

Akce
Chcete-li se této situaci vyhnout, ujistěte se, že ConnectionConsumers používající frontu poskytuje sadu selektorů, které se zabývají všemi zprávami, nebo nastavte továrnu QueueConnectiontak, aby uchovávala zprávy.

Případně prozkoumejte zdroj zprávy.

MQRC_JMS_FORMAT_ERROR (0x93C; 2364)

Příčina
JMS nemůže interpretovat zprávu ve frontě.

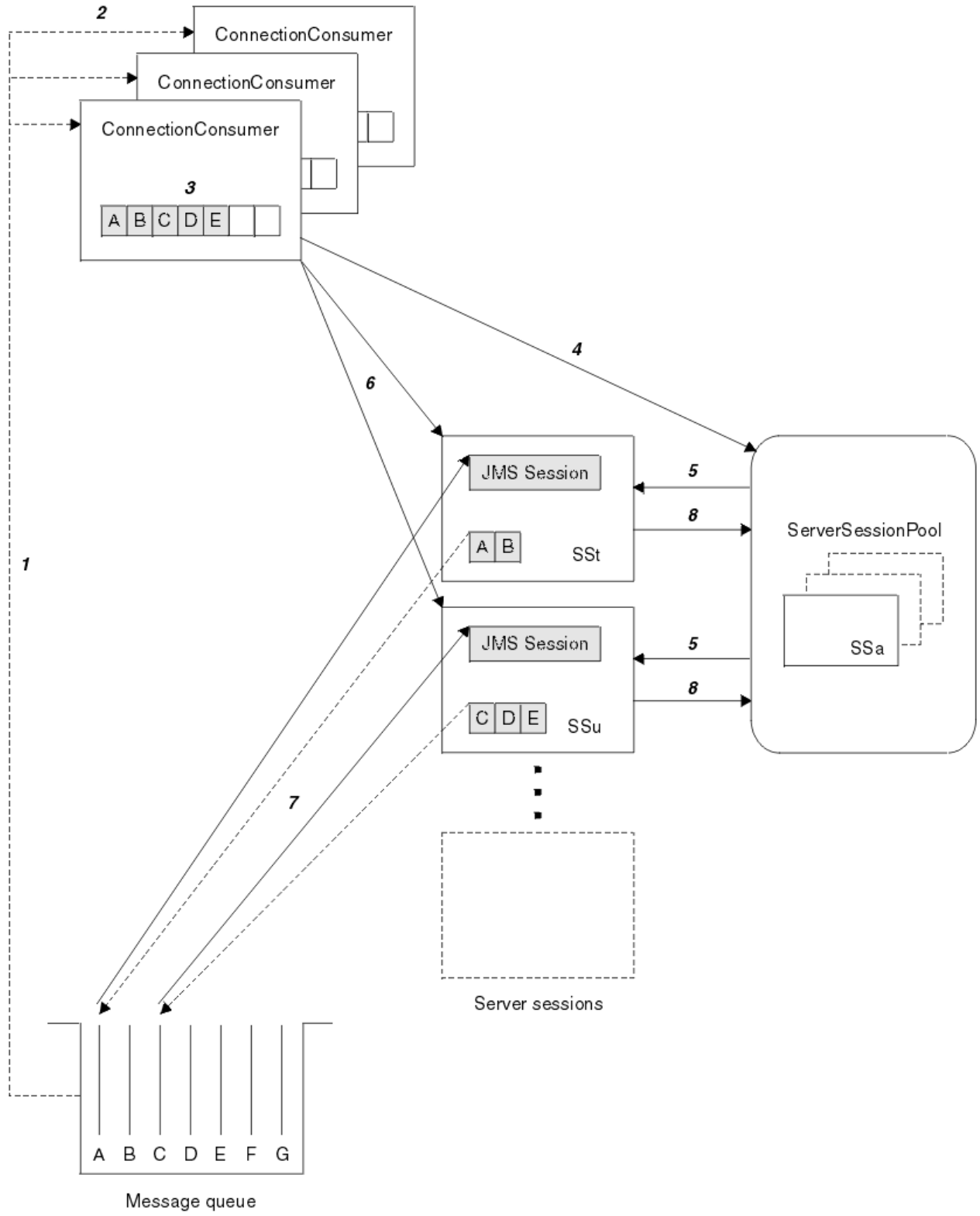
Akce
Prozkoumejte původ zprávy. Produkt JMS obvykle doručuje zprávy v neočekávaném formátu jako BytesMessage nebo TextMessage. Občas se to nezdaří, pokud je zpráva velmi špatně formátovaná.

Další kódy, které se objevují v těchto polích, jsou způsobeny neúspěšným pokusem o vrácení zprávy do fronty vrácení zpět. V této situaci kód popisuje příčinu selhání žadatele. Chcete-li diagnostikovat příčinu těchto chyb, postupujte podle pokynů v části API> Kódy příčiny a dokončení rozhraní API.

Pokud zprávu sestavy nelze vložit do fronty ReplyTo, bude vložena do fronty nedoručených zpráv. V této situaci je pole zpětné vazby deskriptoru MQMD dokončeno podle popisu v tomto tématu. Pole příčiny v MQDLH vysvětluje, proč zprávu sestavy nelze umístit do fronty ReplyTo.

Funkce fondu relací serveru v systému AFS
Toto téma shrnuje funkci fondu relací serveru.

Obrázek 45 na stránce 328 shrnuje principy funkcí ServerSessionPool a ServerSession .



Obrázek 45. Funkce ServerSessionPool a ServerSession

1. 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. ConnectionConsumer vyžaduje jednu nebo více relací ServerSessions z fondu ServerSession.

5. ServerSessions jsou přiděleny z fondu ServerSession.
6. ConnectionConsumer přiřadí odkazy na zprávy k ServerSessions a spustí spuštěné podprocesy ServerSession .
7. Každá relace ServerSession načte své odkazované zprávy z fronty. Předává je do metody onMessage z modulu MessageListener , který je přidružen k relaci JMS .
8. Po dokončení zpracování se relace ServerSession vrátí do fondu.

Aplikační server obvykle poskytuje funkce ServerSessionPool a ServerSession .

z/OS Použití IBM MQ classes for JMS na serveru prostředí JVM CICS Liberty

V operačním systému IBM MQ 9.1.0mohou programy Java spuštěné na serveru CICS Liberty JVM používat IBM MQ classes for JMS pro přístup IBM MQ.

Musíte používat IBM MQ 9.1.0nebo novější verzi adaptéru prostředků IBM MQ . Adaptér prostředků můžete získat z adresáře Fix Central (viz “Instalace adaptéru prostředků v adresáři Liberty” na stránce 432).

V produktu CICS 5.3 jsou k dispozici dvě varianty prostředí JVM produktu Liberty Profile a novější, typy připojení k produktu IBM MQ jsou omezeny následujícím způsobem:

CICS Liberty Standardní

- Adaptér prostředků IBM MQ se může připojit k libovolné verzi produktu IBM MQ ve službě v režimu CLIENT
- Adaptér prostředků IBM MQ se může připojit k libovolné verzi produktu IBM MQ for z/OS v režimu BINDINGS v případě, že neexistuje žádné připojení CICS (aktivní definice prostředku CICS MQCONN) ke stejnému správci front ze stejné oblasti CICS .

CICS Liberty Integrované

- Adaptér prostředků IBM MQ se může připojit k libovolné provozní verzi produktu IBM MQ v režimu KLIENT.
- Připojení v režimu BINDINGS není podporováno.

Podrobnosti o nastavení a konfiguraci systému viz Použití IBM MQ classes for JMS na serveru Liberty JVM v dokumentaci CICS .

Použití IBM MQ classes for JMS/ Jakarta Messaging v IMS

Podpora systému zpráv založená na standardech v prostředí IMS je poskytována prostřednictvím produktu IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging.

Zkontrolujte systémové požadavky pro systém IMS , který používá váš podnik. Další informace viz IMS 15.2 .

Tato sada témat popisuje, jak nastavit IBM MQ classes for JMS v prostředí IMS a omezení rozhraní API, která platí při použití klasických (JMS 1.1) a zjednodušených (JMS 2.0) rozhraní. Seznam informací specifických pro rozhraní API naleznete v části “JMS Omezení rozhraní API” na stránce 334 .

Poznámka: Podobná omezení platí pro starší (JMS 1.0.2) rozhraní specifická pro doménu, ale nejsou zde výslovně popsána.

V 9.3.0 JM 3.0 V 9.3.0 Od IBM MQ 9.3.0, Jakarta Messaging 3.0 je podporován pro vývoj nových aplikací. Produkt IBM MQ 9.3.0 nadále podporuje produkt JMS 2.0 pro existující aplikace. Použití rozhraní API Jakarta Messaging 3.0 a rozhraní API JMS 2.0 ve stejné aplikaci není podporováno. Další informace naleznete v tématu Použití tříd IBM MQ pro systém zpráv JMS/Jakarta.

Podporované IMS závislé oblasti

Podporovány jsou následující závislé typy oblastí:

- MPR
- BMP
- Strategické plánování zařízení
- JMP 31 a 64bitové prostředí JVM (Java Virtual Machine)
- JBP 31 a 64bitová prostředí JVM

Pokud to není výslovně uvedeno v následujících tématech, IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging se chovají stejně ve všech typech oblastí.

Podporované virtuální počítače Java

Produkty IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging vyžadují prostředí IBM Runtime Environment, Java Technology Edition 8. IBM Semeru Runtime Certified Edition pro z/OS, verze 11 není podporována.

Další omezení

Při použití produktu IBM MQ classes for JMS v prostředí IMS platí následující omezení:

- Připojení v režimu klienta nejsou podporována.
- Připojení jsou podporována pouze pro správce front IBM MQ 8.0 používající IBM MQ poskytovatele systému zpráv `Normal`v režimu.

  Atribut **`PROVIDERVERSION`** v továrně připojení musí být buď neurčený, nebo hodnota větší nebo rovna sedmi.
- Použití žádné z továren připojení XA, například `com.ibm.mq.jms.MQXAConnectionFactory`, není podporováno.

Související úlohy

Definování IBM MQ do IMS

Nastavení adaptéru IMS pro použití s IBM MQ classes for JMS/Jakarta Messaging

IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging používají stejný adaptér IBM MQ-IMS , který používají jiné programovací jazyky. Tento adaptér používá prostředek IMS External Subsystem Attach Facility (ESAF).

Než začnete

Před provedením následujícího postupu musíte nakonfigurovat adaptér IMS pro příslušné správce front a řídicí a závislé oblasti systému IMS , jak je popsáno v tématu Nastavení adaptéru IMS.

Upozornění: Nemusíte provádět krok, který popisuje sestavení dynamického stubu, pokud dynamický stub nepotřebujete pro jiné účely.

Po nakonfigurování adaptéru IMS proveďte následující postup.

Postup

1. Aktualizujte proměnnou LIBPATH ve členu IMS PROCLIB, na který odkazuje parametr ENVIRON v JCL závislé oblasti (například DFSJVMEV), tak, aby obsahovala nativní knihovny IBM MQ classes for JMS .

   To znamená adresář zFS , který obsahuje `libmqjims.so`. Například DFSJVMEV může vypadat následovně, kde poslední řádek je adresář obsahující nativní knihovny IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging :

```
LIBPATH=>
/java/latest/bin/j9vm:>
/java/latest/bin:>
/ims/latest/dbdc/imsjava/classic/lib:>
```

```
/ims/latest/dbdc/imsjava/lib:>
/mqm/latest/java/lib
```

2. Přidejte IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging do cesty ke třídám prostředí JVM, kterou používá vaše závislá oblast IMS , aktualizací volby `java.class.path` .

   Postupujte podle pokynů v části Člen DFSJVMMS datové sady IMS PROCLIB.

   Můžete například použít následující, kde řádek tučně označuje aktualizaci:

V 9.3.0 JM 3.0 V 9.3.0

```
-Djava.class.path=/ims/latest/dbdc/imsjava/imsutm.jar:/ims/latest/dbdc/imsjava/imsudb.jar:
/mqm/latest/java/lib/com.ibm.mq.jakarta.client.jar
```

JMS 2.0

```
-Djava.class.path=/ims/latest/dbdc/imsjava/imsutm.jar:/ims/latest/dbdc/imsjava/imsudb.jar:
/mqm/latest/java/lib/com.ibm.mq.allclient.jar
```

   Poznámka: I když je v adresáři obsahujícím soubor IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messagingk dispozici mnoho různých souborů JAR, potřebujete pouze `com.ibm.mq.allclient.jar` (JMS 2.0) nebo `com.ibm.mq.jakarta.client.jar` (Jakarta Messaging 3.0).

3. Zastavte a restartujte všechny IMS závislé oblasti, které budou používat IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging.

Jak pokračovat dále

Vytvořte a nakonfigurujte továrny připojení a místa určení.

Existují tři možné přístupy pro vytvoření instancí implementací IBM MQ továren připojení a míst určení. Podrobnosti viz “Vytvoření a konfigurace továren připojení a míst určení” na stránce 198.

Všimněte si, že všechny tyto tři přístupy jsou platné v prostředí IMS .

Související pojmy

Nastavení adaptéru IMS

Definování IBM MQ do IMS

Transakční chování

Zprávy odeslané a přijaté produktem 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 potvrzení nebo odvolání v instanci objektu `com.ibm.ims.dli.tm.Transaction` nebo normálním ukončením úlohy IMS . V takovém případě je jednotka UOW implicitně potvrzena. Pokud úloha IMS skončí abnormálně, 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()`

Existuje jedna výjimka z tohoto chování. Pokud je relace nebo kontext JMS vytvořen pomocí jednoho z následujících mechanismů:

- `Connection.createSession(false, Session.AUTO_ACKNOWLEDGE)`
- `Connection.createSession(Session.AUTO_ACKNOWLEDGE)`
- `ConnectionFactory.createContext(JMSContext.AUTO_ACKNOWLEDGE)`

pak chování této relace nebo kontextu JMS je následující:

- Všechny odeslané zprávy jsou odeslány mimo jednotku IMS UOW. To znamená, že budou okamžitě k dispozici na cílovém místě určení nebo po dokončení poskytnutého intervalu prodlevy doručení.
- Všechny dočasné zprávy budou přijaty mimo pracovní jednotku IMS UOW, pokud nebyla v továrně připojení, která vytvořila relaci nebo kontext JMS , zadána vlastnost SYNCPOINTALLGETS .
- Trvalé zprávy budou vždy přijímány uvnitř IMS UOW.

To může být užitečné například v případě, že chcete zapsat zprávu auditu do fronty, i když se UOW odvolá.

Důsledky IMS synchronizovaných bodů

Produkt IBM MQ classes for JMS staví na existující podpoře adaptéru IBM MQ , která využívá rozhraní ESAF. To znamená, že se použije zdokumentované chování, včetně všech otevřených popisovačů zavřených adaptérem IMS při výskytu synchronizačního bodu.

Další informace viz “Synchronizační body v aplikacích IMS” na stránce 67.

Pro ilustraci tohoto bodu zvažte spuštění následujícího kódu v prostředí JMP.
Druhé volání funkce `mp.send()` má za následek `JMSException` , protože kód `messageQueue.getUnique(inputMessage)` má za následek zavření všech otevřených připojení IBM MQ a popisovačů objektů.

Podobné chování je pozorováno, pokud bylo volání `getUnique()` nahrazeno voláním `Transaction.commit()`, ale ne, pokud bylo použito volání `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 pro použití v tomto scénáři je následující. V tomto případě je připojení k produktu IBM MQ uzavřeno před voláním funkce `getUnique()`. Připojení a relace se pak znovu vytvoří, aby se odeslala další zpráva.

```
//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

Musíte mít na paměti následující omezení. Pro každého správce front můžete mít pouze jeden manipulátor připojení. Existují důsledky v interakci s produktem IBM MQ při použití jak kódu JMS , tak nativního kódu. Existují omezení pro ověření a autorizaci připojení.

Jeden manipulátor připojení pro každého správce front

V závislých oblastech IMS je v daném okamžiku povolen pouze jeden manipulátor připojení ke specifickému správci front. Všechny následné pokusy o připojení ke stejnému správci front znovu použijí existující manipulátor.

I když by toto chování nemělo způsobit žádné problémy v aplikaci, která používá pouze IBM MQ classes for JMS, může toto chování způsobit problémy v aplikacích, které interagují s produktem IBM MQ, když používají jak IBM MQ classes for JMS , tak MQI v nativním kódu napsaném v jazycích, jako např. COBOL nebo C.

Důsledky interakce s produktem IBM MQ při použití JMS i nativního kódu

K problémům může dojít při prokládání kódu Java a nativního kódu, které používají funkčnost produktu IBM MQ , a když není připojení k produktu IBM MQ uzavřeno před opuštěním nativního kódu nebo kódu jazyka Java.

Například v následujícím pseudokódu je manipulátor připojení ke správci front původně vytvořen v kódu Java pomocí konzoly IBM MQ classes for JMS. Manipulátor připojení je znovu použit v kódu COBOL a zneplatněn voláním MQDISC.

Při příštím použití manipulátoru připojení produktem IBM MQ classes for JMS bude použita hodnota `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 jiné podobné vzory použití, které mohou vést k chybě MQRC_HCONN_ERROR.

Ačkoli je možné sdílet manipulátory připojení IBM MQ mezi nativním kódem a kódem Java (například předchozí příklad by fungoval, kdyby nebylo volání MQDISC) obecně, doporučeným postupem je zavřít všechny manipulátory připojení před změnou z Java na nativní kód, nebo naopak.

Ověření a autorizace připojení

Specifikace JMS umožňuje zadat jméno uživatele a heslo pro ověření a autorizaci při vytváření připojení nebo objektu kontextu JMS .

Toto není podporováno v prostředí IMS . Při pokusu o vytvoření připojení při zadávání jména uživatele a hesla dojde k vyvolání `JMS Exception` . Při pokusu o vytvoření kontextu JMS při zadávání jména uživatele a hesla dojde k vyvolání `JMSRuntimeException` .

Místo toho musí být použity existující mechanizmy pro ověření a autorizaci při připojování k produktu IBM MQ z prostředí IMS .

Další informace viz Nastavení zabezpečení v systému z/OS. Zvláště pak téma ID uživatelů pro kontrolu zabezpečení, které popisuje použitelná ID uživatelů.

Související úlohy

Nastavení zabezpečení na systému z/OS

JMS Omezení rozhraní API

Z perspektivy specifikace JMS se IBM MQ classes for JMS zachází IMS jako s aplikačním serverem kompatibilním s Java EE nebo Jakarta EE , který má vždy probíhající transakci JTA.

Například nikdy nemůžete volat `javax.jms.Session.commit()` v IMS, protože specifikace JMS uvádí, že ji nemůžete volat v EJB JEE nebo ve webovém kontejneru, zatímco probíhá transakce JTA.

To má za následek následující omezení rozhraní API JMS , kromě omezení popsaných v části “Transakční chování” na stránce 331.

Omezení klasického rozhraní API

- `javax.jms.Connection.createConnectionConsumer(javax.jms.Destination, String, javax.jms.ServerSessionPool, int)` vždy vyvolá výjimku `JMSException`.
- `javax.jms.Connection.createDurableConnectionConsumer(javax.jms.Topic, String, String, javax.jms.ServerSessionPool, int)` vždy vyvolá výjimku `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 vyvolá výjimku `JMSException`.
- `javax.jms.Connection.createSharedDurableConnectionConsumer(javax.jms.Topic, String, String, javax.jms.ServerSessionPool, int)` vždy vyvolá výjimku `JMSException`.
- `javax.jms.Connection.setClientID()` vždy vyvolá výjimku `JMSException`.
- `javax.jms.Connection.setExceptionListener(javax.jms.ExceptionListener)` vždy vyvolá výjimku `JMSException`.
- `javax.jms.Connection.stop()` vždy vyvolá výjimku `JMSException`.
- `javax.jms.MessageConsumer.setMessageListener(javax.jms.MessageListener)` vždy vyvolá výjimku `JMSException`.
- `javax.jms.MessageConsumer.getMessageListener()` vždy vyvolá výjimku `JMSException`.
- `javax.jms.MessageProducer.send(javax.jms.Destination, javax.jms.Message,javax.jms.CompletionListener)` vždy vyvolá výjimku `JMSException`.
- `javax.jms.MessageProducer.send(javax.jms.Destination, javax.jms.Message, int, int, long, javax.jms.CompletionListener)` vždy vyvolá výjimku `JMSException`.

- `javax.jms.MessageProducer.send(javax.jms.Message, int, int, long, javax.jms.CompletionListener)` vždy vyvolá výjimku `JMSException`.
- `javax.jms.MessageProducer.send(javax.jms.Message,javax.jms.CompletionListener )` vždy vyvolá výjimku `JMSException`.
- `javax.jms.Session.run()` vždy vyvolá výjimku `JMSRuntimeException`.
- `javax.jms.Session.setMessageListener(javax.jms.MessageListener)` vždy vyvolá výjimku `JMSException`.
- `javax.jms.Session.getMessageListener()` vždy vyvolá výjimku `JMSException`.

Zjednodušená omezení rozhraní API

- `javax.jms.JMSContext.createContext(int)` vždy vyvolá výjimku `JMSRuntimeException`.
- `javax.jms.JMSContext.setClientID(String)` vždy vyvolá výjimku `JMSRuntimeException`.
- `javax.jms.JMSContext.setExceptionListener(javax.jms.ExceptionListener)` vždy vyvolá výjimku `JMSRuntimeException`.
- `javax.jms.JMSContext.stop()` vždy vyvolá výjimku `JMSRuntimeException`.
- `javax.jms.JMSProducer.setAsync(javax.jms.CompletionListener)` vždy vyvolá výjimku `JMSRuntimeException`.

Použití produktu IBM MQ classes for Java

Použijte IBM MQ v prostředí Java . IBM MQ classes for Java Povolit aplikaci Java připojit se k produktu IBM MQ jako klient IBM MQ nebo se připojit přímo ke správci front IBM MQ .

Poznámka:

Stabilized Produkt IBM neprovede žádná další vylepšení produktu IBM MQ classes for Java a jsou funkčně stabilizovány na úrovni dodané v produktu IBM MQ 8.0. Existující aplikace, které používají produkt IBM MQ classes for Java , jsou i nadále plně podporovány, ale nové funkce nebudou přidány a požadavky na vylepšení budou odmítnuty. Plně podporováno znamená, že defekty budou opraveny společně se změnami, které jsou vynucené změnami systémových požadavků IBM MQ .

IBM MQ classes for Java nejsou v produktu IMSpodporovány.

IBM MQ classes for Java nejsou v produktu WebSphere Libertypodporovány. Nesmí být použity s funkcí systému zpráv IBM MQ Liberty ani s generickou podporou systému JCA . Další informace naleznete v tématu Použití rozhraní Java WebSphere MQ v prostředí J2EE/JEE.

IBM MQ classes for Java je jedno ze tří alternativních rozhraní API, která mohou aplikace Java použít pro přístup k prostředkům IBM MQ . Další rozhraní API jsou:

- V 9.3.0 JM 3.0 V 9.3.0 IBM MQ classes for Jakarta Messaging
- JMS 2.0 IBM MQ classes for JMS

Další informace viz téma “Přístup k IBM MQ z Java -Volba rozhraní API” na stránce 82.

V produktu IBM MQ 9.3jsou IBM MQ classes for Java sestaveny pomocí Java 8. Běhové prostředí Java 8 podporuje spouštění starších verzí souborů tříd.

IBM MQ classes for Java zapouzdřit rozhraní MQI (Message Queue Interface), nativní rozhraní API systému IBM MQ a použít podobný objektový model pro rozhraní C++ a .NET pro IBM MQ.

Programovatelné volby umožňují produktu IBM MQ classes for Java připojit se k produktu IBM MQ jedním z následujících způsobů:

- V klientském režimu jako IBM MQ MQI client pomocí protokolu TCP/IP (Transmission Control Protocol/ Internet Protocol).
- V režimu vazebse přímé připojení 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 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 ve stejném systému, ve kterém je spuštěn správce front, nebo v jiném systému. V každém případě se produkt IBM MQ classes for Java připojí ke správci front prostřednictvím protokolu TCP/IP.

Další informace o zápisu aplikací pro použití připojení v režimu klienta naleznete v tématu “IBM MQ classes for Java režimy připojení” na stránce 360.

Připojení v režimu vazeb

Při použití v režimu vazeb produkt IBM MQ classes for Java používá rozhraní JNI (Java Native Interface) k přímému volání do existujícího rozhraní API správce front, nikoli ke komunikaci prostřednictvím sítě. Ve většině prostředí poskytuje připojení v režimu vazeb lepší výkon pro aplikace IBM MQ classes for Java než připojení v režimu klienta, a to tím, že se vyhnete nákladům na komunikaci TCP/IP.

Aplikace, které používají IBM MQ classes for Java pro připojení v režimu vazeb, musí být spuštěny ve stejném systému jako správce front, ke kterému se připojují.

Běhové prostředí Java , které se používá ke spuštění aplikace IBM MQ classes for Java , musí být nakonfigurováno pro načtení knihoven IBM MQ classes for Java ; další informace viz “IBM MQ classes for Java knihovny” na stránce 345 .

Další informace o způsobu zápisu aplikací pro použití připojení v režimu vazeb naleznete v části “IBM MQ classes for Java režimy připojení” na stránce 360.

Související pojmy

Jazyková rozhraní produktu IBM MQ Java

“Použití IBM MQ classes for JMS/Jakarta Messaging” na stránce 78
IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging jsou poskytovatelé systému zpráv Java dodávaní s produktem IBM MQ. Kromě implementace rozhraní definovaných ve specifikacích JMS a Jakarta Messaging tito poskytovatelé systému zpráv přidávají do rozhraní API systému zpráv Java dvě sady rozšíření.

Související úlohy

Trasování aplikací IBM MQ classes for Java

Odstraňování problémů Java a JMS

Proč bych měl používat IBM MQ classes for Java?

Aplikace Java může použít buď IBM MQ classes for Java , nebo IBM MQ classes for JMS pro přístup k prostředkům IBM MQ .

Poznámka: V 9.3.0 V 9.3.0 Ačkoli existující aplikace, které používají IBM MQ classes for Java , jsou i nadále plně podporovány, nové aplikace by měly používat IBM MQ classes for Jakarta Messaging. Funkce, které byly nedávno přidány do produktu IBM MQ, jako např. asynchronní spotřeba a automatické opětovné připojení, nejsou k dispozici v produktu IBM MQ classes for Java, ale jsou k dispozici v IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging. Další informace naleznete v tématech “Proč bych měl používat IBM MQ classes for JMS?” na stránce 80 a “Proč bych měl používat IBM MQ classes for Jakarta Messaging?” na stránce 79.

Poznámka:

Stabilized Produkt IBM neprovede žádná další vylepšení produktu IBM MQ classes for Java a jsou funkčně stabilizovány na úrovni dodané v produktu IBM MQ 8.0. Existující aplikace, které používají

produkt IBM MQ classes for Java , jsou i nadále plně podporovány, ale nové funkce nebudou přidány a požadavky na vylepšení budou odmítnuty. Plně podporováno znamená, že defekty budou opraveny společně se změnami, které jsou vynucené změnami systémových požadavků IBM MQ .

IBM MQ classes for Java nejsou v produktu IMSpodporovány.

IBM MQ classes for Java nejsou v produktu WebSphere Libertypodporovány. Nesmí být použity s funkcí systému zpráv IBM MQ Liberty ani s generickou podporou systému JCA . Další informace naleznete v tématu Použití rozhraní Java WebSphere MQ v prostředí J2EE/JEE.

Související pojmy

“Přístup k IBM MQ z Java -Volba rozhraní API” na stránce 82
Produkt IBM MQ poskytuje tři jazyková rozhraní Java .

Předpoklady pro IBM MQ classes for Java

Chcete-li používat produkt IBM MQ classes for Java, potřebujete určité 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 JDK (Java Development Kit). Podrobné informace o JDK podporovaných vaším operačním systémem naleznete v informacích Systémové požadavky pro IBM MQ .

Chcete-li spustit aplikace IBM MQ classes for Java , potřebujete následující softwarové komponenty:

- Správce front IBM MQ pro aplikace, které se připojují ke správci front.
- Prostředí JRE (Java Runtime Environment) 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, z/OS UNIX System Services (z/OS UNIX)

Pokud požadujete, aby připojení TLS používala šifrovací moduly s certifikací FIPS 140-2, potřebujete poskytovatele JSSE FIPS IBM Java (IBMJSSEFIPS). Každý produkt IBM JDK a JRE ve verzi 1.4.2 nebo novější obsahuje produkt IBMJSSEFIPS.

Adresy Internet Protocol verze 6 (IPv6) můžete použít ve svých IBM MQ classes for Java aplikacích , pokud IPv6 je podporován vaším prostředím JVM (Java Virtual Machine) a implementací TCP/IP ve vašem operačním systému.

Spuštění aplikací IBM MQ classes for Java v rámci Java EE

Před použitím souboru IBM MQ classes for Java v souboru Java EEje třeba vzít v úvahu určitá omezení a aspekty návrhu.

Produkt IBM MQ classes for Java má omezení při použití v prostředí Java Platform, Enterprise Edition (Java EE). Existují také další aspekty, které je třeba vzít v úvahu při návrhu, implementaci a správě aplikace IBM MQ classes for Java , která běží v prostředí Java EE . Tato omezení a aspekty jsou popsány v následujících oddílech.

Omezení transakcí JTA

Jediný podporovaný správce transakcí pro aplikace používající IBM MQ classes for Java je IBM MQ sám o sobě. Ačkoli aplikace pod řízením JTA může používat produkt IBM MQ classes for Java, jakákoli práce prováděná prostřednictvím těchto tříd není řízena pracovními jednotkami JTA. Místo toho vytvářejí lokální pracovní jednotky oddělené od jednotek spravovaných aplikačním serverem prostřednictvím rozhraní JTA. Zejména jakékoli odvolání transakce JTA nevede k odvolání žádných odeslaných nebo přijatých zpráv. Toto omezení platí pro transakce spravované aplikací nebo objekty bean a pro transakce spravované kontejnerem a všechny kontejnery Java EE . Chcete-li provádět práci se systémem zpráv

přímo s produktem IBM MQ v rámci transakcí koordinovaných aplikačním serverem, musíte místo toho použít IBM MQ classes for JMS .

Vytvoření podprocesu

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 provedena v podprocesu 'worker' vytvořeném interně produktem IBM MQ classes for Java. Další podprocesy lze vytvářet interně, například za účelem vymazání nepoužívaných připojení z fondu připojení nebo odebrání odběrů pro ukončené aplikace publikování/odběru.

Některé aplikace systému Java EE (například aplikace spuštěné v kontejnerech EJB a ve webových kontejnerech) nesmí vytvářet nové podprocesy. Místo toho musí být veškerá práce provedena na hlavních podprocesech aplikace spravovaných aplikačním serverem. Když aplikace používají produkt IBM MQ classes for Java, aplikační server nemusí být schopen rozlišit 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 prostředí Java EE neporušuje, a proto je lze použít.

Bezpečnostní omezení

Zásady zabezpečení implementované aplikačním serverem mohou bránit určitým operacím prováděným rozhraním API IBM MQ classes for Java , například vytváření a provozování nových řídicích podprocesů (jak je popsáno v předchozích sekcích).

Například aplikační servery obvykle standardně běží se zakázaným zabezpečením Java a umožňují jeho povolení prostřednictvím některé konfigurace specifické pro aplikační server (některé aplikační servery také umožňují podrobnější konfiguraci zásad používaných v rámci zabezpečení produktu Java). Když je povoleno Java Zabezpečení, IBM MQ classes for Java může porušit pravidla pro podprocesy zásad zabezpečení Java definovaná pro aplikační server a rozhraní API nemusí být schopné vytvořit všechny podprocesy, které potřebuje, aby fungovalo. Chcete-li zabránit problémům se správou podprocesů, použití produktu IBM MQ classes for Java není podporováno v prostředích, kde je povoleno zabezpečení produktu Java .

Aspekty izolace aplikací

Zamýšlenou výhodou spouštění aplikací v prostředí Java EE je izolace aplikací. Návrh a implementace produktu IBM MQ classes for Java před datem prostředí Java EE . Produkt IBM MQ classes for Java lze použít způsobem, který nepodporuje koncepci izolace aplikace. Konkrétní příklady úvah v této oblasti zahrnují:

- Použití statických nastavení (v rámci celého procesu prostředí JVM) v rámci třídy MQEnvironment, například:
  - ID uživatele a heslo, které se mají použít pro identifikaci a ověření připojení.
  - název hostitele, port a kanál použité pro připojení klienta
  - Konfigurace TLS pro zabezpečená připojení klienta

  Úprava vlastností prostředí MQEnvironment ve prospěch jedné aplikace má vliv i na ostatní aplikace používající stejné vlastnosti. Při spuštění v prostředí s více aplikacemi, jako je například Java EE, musí každá aplikace používat svou vlastní odlišnou konfiguraci prostřednictvím vytvoření objektů MQQueueManager se specifickou sadou vlastností, namísto toho, aby používala vlastnosti nakonfigurované v rámci třídy MQEnvironment pro celý proces.
- Třída MQEnvironment zavádí řadu statických metod, které působí globálně na všechny aplikace používající produkt IBM MQ classes for Java v rámci stejného procesu prostředí JVM, a neexistuje žádný způsob, jak potlačit toto chování pro konkrétní aplikace. Příklady:
  - konfigurace vlastností TLS, například umístění úložiště klíčů
  - konfigurace uživatelských procedur kanálu klienta

  - povolení nebo zakázání diagnostického trasování
  - správa výchozího fondu připojení používaného k optimalizaci použití připojení ke správcům front

  Vyvolání těchto metod ovlivní všechny aplikace spuštěné ve stejném prostředí Java EE .
- Sdružování připojení je povoleno pro optimalizaci procesu vytváření více připojení ke stejnému správci front. Výchozí správce fondu připojení je v rámci celého procesu a je sdílen více aplikacemi. Změny konfigurace fondu připojení, například nahrazení výchozího správce připojení pro jednu aplikaci pomocí metody MQEnvironment.setDefaultConnectionManager(), tedy ovlivní ostatní aplikace spuštěné na stejném aplikačním serveru Java EE.
- Protokol TLS je konfigurován pro aplikace používající produkt IBM MQ classes for Java s použitím vlastností třídy MQEnvironment a objektu MQQueueManager . Není integrován se spravovanou konfigurací zabezpečení samotného aplikačního serveru. Musíte se ujistit, že jste správně nakonfigurovali produkt IBM MQ classes for Java tak, aby poskytoval požadovanou úroveň zabezpečení, a že nepoužíváte konfiguraci aplikačního serveru.

Omezení režimu vazeb

Produkty IBM MQ a WebSphere Application Server lze nainstalovat na stejný počítač tak, aby se hlavní verze správce front a adaptéru prostředků IBM MQ dodávané v produktu WebSphere Application Server lišily.

Pokud se hlavní verze správce front a adaptéru prostředků liší, nelze připojení vazeb použít. Všechna připojení z produktu WebSphere Application Server ke správci front používající adaptér prostředků musí používat připojení typu klienta. Připojení vazeb lze použít, pokud jsou verze stejné.

Převody znakových řetězců v souboru IBM MQ classes for Java

IBM MQ classes for Java používají CharsetEncoders a CharsetDecoders přímo pro převod znakových řetězců. Výchozí chování pro převod znakových řetězců lze konfigurovat se dvěma systémovými vlastnostmi. Zpracování zpráv, které obsahují nemapovatelné znaky, lze konfigurovat prostřednictvím produktu `com.ibm.mq.MQMD`.

Před IBM MQ 8.0byly převody řetězců v souboru IBM MQ classes for Java provedeny voláním metod `java.nio.charset.Charset.decode(ByteBuffer)` a `Charset.encode(CharBuffer)` .

Použití jedné z těchto metod má za následek výchozí náhradu (REPLACE) chybných nebo nepřeložitelných dat. Toto chování může zakrývat chyby v aplikacích a vést k neočekávaným znakům, například ?, v přeložených datech.

V systému IBM MQ 8.0lze tyto problémy zjišťovat dříve a efektivněji pomocí IBM MQ classes for Java přímo používat CharsetEncoders a CharsetDecoders a explicitně konfigurovat zpracování chybných a nepřeložitelných dat. Výchozí chování je REPORT takové problémy vyvoláním vhodného `MQException`.

Konfigurace

Překlad z UTF-16 (reprezentace znaků používaná v produktu Java) do nativní znakové sady, například UTF-8, se nazývá *kódování*, zatímco překlad v opačném směru se nazývá *dekódování*.

Dekódování má výchozí chování pro `CharsetDecoders`a hlásí chyby vyvoláním výjimky.

Jedno nastavení se používá k určení `java.nio.charset.CodingErrorAction` pro řízení ošetřování chyb při kódování i dekódování. Jedno další nastavení se používá k řízení náhradního bajtu nebo bajtů při kódování. V operacích dekódování bude použit výchozí řetězec náhrady Java .

Konfigurace nepřeložitelného zpracování dat v produktu IBM MQ classes for Java

V produktu IBM MQ 8.0obsahuje produkt `com.ibm.mq.MQMD` následující dvě pole:

byte [] unMappableVýměna
: Posloupnost bajtů, která bude zapsána do kódovaného řetězce, pokud nelze vstupní znak přeložit, a zadali jste REPLACE.

Předvolba: "?".getBytes()
Výchozí řetězec náhrady Java se používá v operacích dekódování.

java.nio.charset.CodingErrorAction unmappableAction
Určuje akci, která má být provedena pro nepřeložitelná data při kódování a dekódování:

Předvolba: CodingErrorAction.REPORT;

Systémové vlastnosti pro nastavení předvoleb systému

V produktu IBM MQ 8.0jsou k dispozici následující dvě systémové vlastnosti Java pro konfiguraci výchozího chování, pokud jde o převod znakového řetězce.

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 získá náhradní bajty, které se mají použít, když znak nelze mapovat v operaci kódování. Výchozí řetězec náhrady Java se používá v operacích dekódování.

Chcete-li se vyhnout záměně znakových a nativních bajtových reprezentací Java , měli byste zadat `com.ibm.mq.cfg.jmqi.UnmappableCharacterReplacement` jako desetinné číslo představující náhradní bajt v nativní znakové sadě.

Například desetinná hodnota ?, jako nativní bajt, je 63, pokud je nativní znaková sada založená na ASCII, jako např. ISO-8859-1, zatímco 111, pokud je nativní znaková sada EBCDIC.

Poznámka: Všimněte si, že pokud má objekt MQMD nebo MQMessage nastavena pole **`unmappableAction`** nebo **`unMappableReplacement`** , mají hodnoty těchto polí přednost před systémovými vlastnostmi Java . To umožňuje v případě potřeby přepsat hodnoty zadané systémovými vlastnostmi Java pro každou zprávu.

Související pojmy
“Převody znakových řetězců v souboru IBM MQ classes for JMS” na stránce 134
IBM MQ classes for JMS používají CharsetEncoders a CharsetDecoders přímo pro převod znakových řetězců. Výchozí chování pro převod znakových řetězců lze konfigurovat se dvěma systémovými vlastnostmi. Zpracování zpráv, které obsahují nemapovatelné znaky, lze konfigurovat pomocí vlastností zprávy pro nastavení akce UnmappableCharactera nahrazujících bajtů.

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 uvádí, jak nakonfigurovat produkt IBM MQ classes for Java po instalaci.

Co je instalováno pro produkt IBM MQ classes for Java

Nejnovější verze produktu IBM MQ classes for Java se instaluje s produktem IBM MQ. Možná budete muset přepsat výchozí volby instalace, abyste se ujistili, že se tak děje.

Další informace o instalaci produktu IBM MQ viz:

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

Soubory IBM MQ classes for Java jsou obsaženy v souborech archivu Java (JAR), `com.ibm.mq.jara` `com.ibm.mq.jmqi.jar`.

Podpora standardních záhlaví zpráv, jako např. PCF (Programmable Command Format), je obsažena v souboru JAR `com.ibm.mq.headers.jar`.

Podpora formátu PCF (Programmable Command Format) je obsažena v souboru JAR `com.ibm.mq.pcf.jar`.

Poznámka: Nedoporučuje se používat IBM MQ classes for Java v rámci aplikačního serveru. Informace o omezeních, která platí při spuštění v tomto prostředí, naleznete v části “Spuštění aplikací IBM MQ classes for Java v rámci Java EE” na stránce 337. Další informace naleznete v tématu Použití rozhraní WebSphere MQ Java v prostředí J2EE/JEE.

Důležité: Kromě přemístitelných souborů JAR popsaných v tématu “IBM MQ classes for Java přemístitelné soubory JAR” na stránce 341není podporováno kopírování souborů JAR IBM MQ classes for Java nebo nativních knihoven do jiných počítačů nebo do jiného umístění na počítači, kde byl nainstalován produkt IBM MQ classes for Java .

IBM MQ classes for Java přemístitelné soubory JAR
Přemístitelné soubory JAR lze přesunout do systémů, které potřebují spustit IBM MQ classes for Java.

Důležité:

- Kromě relokovatelných souborů JAR popsaných v části Relokovatelné soubory JARnení podporováno kopírování souborů JAR IBM MQ classes for Java nebo nativních knihoven do jiných počítačů nebo do jiného umístění na počítači, kde byl nainstalován produkt IBM MQ classes for Java .
- Chcete-li se vyhnout konfliktům zavaděče tříd, nedoporučuje se zabalit přemístitelné soubory JAR do více aplikací ve stejném běhovém prostředí Java . V tomto scénáři zvažte zpřístupnění přemísťovatelných souborů JAR IBM MQ na cestě ke třídě běhového prostředí Java .
- Nezahrnujte přemístitelné soubory JAR do aplikací implementovaných do aplikačních serverů Java EE , jako např. WebSphere Application Server. V těchto prostředích by měl být implementován a použit adaptér prostředků IBM MQ , protože obsahuje soubor IBM MQ classes for Java. Všimněte si, že produkt WebSphere Application Server vkládá adaptér prostředků IBM MQ , takže jej není nutné ručně implementovat do tohoto prostředí. Kromě toho nejsou v produktu WebSphere Libertypodporovány položky IBM MQ classes for Java . Další informace viz “Liberty a adaptér prostředků IBM MQ” na stránce 428.
- Pokud ve svých aplikacích sdružujete přemístitelné soubory JAR, ujistěte se, že jste zahrnuli všechny nezbytné soubory JAR, jak je popsáno v tématu Přemísťitelné soubory JAR. Měli byste se také ujistit, že máte odpovídající procedury pro aktualizaci souborů JAR v balíku v rámci údržby aplikace, abyste se ujistili, že produkt IBM MQ classes for Java zůstane aktuální a známé problémy budou znovu mediovány.

Přemístitelné soubory JAR

V rámci podniku lze následující soubory přesunout do systémů, které vyžadují spuštění aplikací IBM MQ classes for Java :

- JMS 2.0 `com.ibm.mq.allclient.jar` “1” na stránce 342
- V 9.3.0 JM 3.0 V 9.3.0 `com.ibm.mq.jakarta.client.jar` “2” na stránce 342
- V 9.3.3 Removed `com.ibm.mq.traceControl.jar`
- V 9.3.5 `bcpkix-jdk18on.jar` “3” na stránce 342
- V 9.3.0 V 9.3.0 `bcpkix-jdk15to18.jar` “4” na stránce 342
- V 9.3.5 `bcprov-jdk18on.jar` “3” na stránce 342
- V 9.3.0 V 9.3.0 `bcprov-jdk15to18.jar` “4” na stránce 342
- V 9.3.5 `bcutil-jdk18on.jar` “3” na stránce 342
- V 9.3.0 V 9.3.0 `bcutil-jdk15to18.jar` “4” na stránce 342
- V 9.3.3 `jackson-annotations.jar`
- V 9.3.3 `jackson-core.jar`
- V 9.3.3 `jackson-databind.jar`

- `org.json.jar`

Notes:

1. JMS 2.0 a JMS 1.1
2. Jakarta Messaging 3.0
3. Continuous DeliveryOdIBM MQ 9.3.5
4. Long Term Support a Continuous Delivery před IBM MQ 9.3.5

Poskytovatel zabezpečení Bouncy Castle a CMS podporují soubory JAR

Poskytovatel zabezpečení Bouncy Castle a soubory JAR podpory CMS jsou povinné. Další informace viz Podpora jiných prostředí JRE nežIBM s AMS.

V 9.3.5 Pro Continuous Delivery z IBM MQ 9.3.5jsou nezbytné následující soubory JAR:

- `bcpkix-jdk18on.jar`
- `bcprov-jdk18on.jar`
- `bcutil-jdk18on.jar`

LTS V případě souborů Long Term Support a Continuous Delivery před IBM MQ 9.3.5jsou vyžadovány následující soubory JAR:

- `bcpkix-jdk15to18.jar`
- `bcprov-jdk15to18.jar`
- `bcutil-jdk15to18.jar`

org.json.jar

Soubor `org.json.jar` je povinný, pokud vaše aplikace IBM MQ classes for Java používá CCDT ve formátu JSON.

com.ibm.mq.allclient.jar a `com.ibm.mq.jakarta.client.jar`

Soubory `com.ibm.mq.allclient.jar` a `com.ibm.mq.jakarta.client.jar` obsahují třídy IBM MQ classes for JMS, IBM MQ classes for Javaa PCF a záhlaví. Přesunete-li tyto soubory do nového umístění, ujistěte se, že jste provedli kroky k udržení tohoto nového umístění s novými opravnými sadami IBM MQ Fix Pack. Také se ujistěte, že použití souborů je známo podpoře IBM , pokud dostáváte prozatímní opravu.

Chcete-li určit verzi souboru `com.ibm.mq.allclient.jar` nebo souboru `com.ibm.mq.jakarta.client.jar` , použijte následující příkaz:

V 9.3.0 JM 3.0 V 9.3.0

```
java -jar com.ibm.mq.jakarta.client.jar
```

JMS 2.0

```
java -jar com.ibm.mq.allclient.jar
```

Následující příklad ukazuje ukázku výstupu z tohoto příkazu:

```
C:\Program Files\IBM\MQ_1\java\lib>java -jar com.ibm.mq.allclient.jar
Name:       Java Message Service Client
Version:    9.3.0.0
Level:      p000-L140428.1
Build Type: Production
Location:   file:/C:/Program Files/IBM/MQ_1/java/lib/com.ibm.mq.allclient.jar

Name:       IBM MQ classes for Java Message Service
Version:    9.3.0.0
Level:      p000-L140428.1
```

```
Build Type: Production
Location:   file:/C:/Program Files/IBM/MQ_1/java/lib/com.ibm.mq.allclient.jar

Name:       IBM MQ JMS Provider
Version:    9.3.0.0
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:    9.3.0.0
Level:      p000-L140428.1
Build Type: Production
Location:   file:/C:/Program Files/IBM/MQ_1/java/lib/com.ibm.mq.allclient.jar
```

jackson-annotations.jar, jackson-core.jar a jackson-databind.jar

V 9.3.3

Tři soubory JAR Jackson jsou vyžadovány, pokud vaše aplikace IBM MQ classes for Java vytvoří zabezpečená připojení TLS ke správci front.

Instalační adresáře pro 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 v závislosti na platformě.

Instalační adresáře pro soubory IBM MQ classes for Java

Tabulka 49 na stránce 343 ukazuje, kde jsou nainstalovány soubory IBM MQ classes for Java .

Tabulka 49. IBM MQ classes for Java instalační adresáře

Platforma	**Adresář**
AIX AIX	*MQ_INSTALLATION_PATH*/java/lib
	/QIBM/ProdData/mqm/java/lib
Linux Linux	*MQ_INSTALLATION_PATH*/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, například programy IVP (Installation Verification Programs), jsou dodávány s produktem IBM MQ. Tabulka 50 na stránce 343 ukazuje, kde jsou nainstalovány ukázkové aplikace. Ukázky IBM MQ classes for Java jsou v podadresáři s názvem `wmqjava`. Ukázky PCF jsou v podadresáři s názvem `pcf`.

Tabulka 50. Adresáře ukázek

Platforma	**Adresář**
AIX AIX	*MQ_INSTALLATION_PATH*/samp/wmqjava/
IBM i IBM i	/QIBM/ProdData/mqm/java/samples
Linux Linux	*MQ_INSTALLATION_PATH*/samp/wmqjava/

Tabulka 50. Adresáře ukázek (pokračování)	
Platforma	**Adresář**
Windows	*MQ_INSTALLATION_PATH*\tools\wmqjava\
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

Prostředí IBM MQ classes for JMS vyžaduje prostředí Java 7 (nebo vyšší) Java Runtime Environment (JRE). Vhodné prostředí JRE je instalováno s produktem IBM MQ. Tabulka 51 na stránce 344 ukazuje, kde je toto prostředí JRE nainstalováno. Chcete-li spustit programy Java , jako jsou poskytnuté ukázky, pomocí tohoto prostředí JRE buď explicitně vyvolejte *JRE_LOCATION*`/bin/java` , nebo přidejte *JRE_LOCATION*`/bin` do prostředí PATH (nebo ekvivalentní) pro vaši platformu, kde *JRE_LOCATION* je adresář uvedený v Tabulka 51 na stránce 344.

Tabulka 51. Adresáře prostředí JRE	
Platforma	**Adresář**
AIX	*MQ_INSTALLATION_PATH*/java/jre
IBM i	/QIBM/ProdData/mqm/java/jre
Linux	*MQ_INSTALLATION_PATH*/java/jre
Windows	*MQ_INSTALLATION_PATH*\java\jre
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 IBM MQ classes for Java
Chcete-li spustit aplikace IBM MQ classes for Java , musí jejich cesta ke třídám obsahovat adresáře IBM MQ classes for Java a ukázky.

Aby mohly být spuštěny aplikace IBM MQ classes for Java , musí jejich cesta ke třídám obsahovat příslušný adresář IBM MQ classes for Java . Chcete-li spustit ukázkové aplikace, musí cesta ke třídám také obsahovat příslušné ukázkové adresáře. Tyto informace lze zadat v příkazu vyvolání Java nebo v proměnné prostředí **CLASSPATH** .

Důležité: Nastavení Java volby `-Xbootclasspath` tak, aby zahrnovala IBM MQ classes for Java , není podporováno.

Tabulka 52 na stránce 344 zobrazuje odpovídající nastavení **CLASSPATH** , které se má použít na každé platformě ke spuštění aplikací IBM MQ classes for Java , včetně ukázkových aplikací.

Tabulka 52. ***CLASSPATH*** *nastavení ke spuštění aplikací IBM MQ classes for Java , včetně ukázkových aplikací IBM MQ classes for Java*	
Platforma	**CLASSPATH nastavení**
AIX	CLASSPATH= *MQ_INSTALLATION_PATH*/java/lib/com.ibm.mq.jar: *MQ_INSTALLATION_PATH*/samp/wmqjava/samples:

*Tabulka 52. **CLASSPATH** nastavení ke spuštění aplikací IBM MQ classes for Java , včetně ukázkových aplikací IBM MQ classes for Java (pokračování)*	
Platforma	**CLASSPATH nastavení**
IBM i	CLASSPATH=/QIBM/ProdData/mqm/java/lib/com.ibm.mq.jar: /QIBM/ProdData/mqm/java/samples/wmqjava/samples:
Linux	CLASSPATH= *MQ_INSTALLATION_PATH*/java/lib/com.ibm.mq.jar: *MQ_INSTALLATION_PATH*/samp/wmqjava/samples:
Windows	CLASSPATH= *MQ_INSTALLATION_PATH*\Java\lib\com.ibm.mq.jar; *MQ_INSTALLATION_PATH*\tools\wmqjava\samples;
z/OS	CLASSPATH= *MQ_INSTALLATION_PATH*/mqm/V9R3M0/java/lib/com.ibm.mq.jar: *MQ_INSTALLATION_PATH*/mqm/V9R3M0/java/samples/wmqjava: *MQ_INSTALLATION_PATH*/mqm/V9R3M0/java/samples/pcf

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

Pokud kompilujete pomocí volby `-Xlint` , může se vám zobrazit zpráva s varováním, že `com.ibm.mq.ese.jar` není přítomen. Varování můžete ignorovat. Tento soubor je přítomen pouze v případě, že jste nainstalovali produkt Advanced Message Security.

Skripty dodávané 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.

MQ_JAVA_INSTALL_PATH
Tato proměnná prostředí určuje adresář, kde je nainstalován produkt IBM MQ classes for Java , jak je uvedeno v IBM MQ classes for Java instalačních adresářích.

MQ_JAVA_LIB_PATH
Tato proměnná prostředí uvádí adresář, kde jsou uloženy knihovny IBM MQ classes for Java , jak ukazuje Umístění knihoven IBM MQ classes for Java pro každou platformu. Některé skripty dodávané s produktem IBM MQ classes for Java, například IVTRun, používají tuto proměnnou prostředí.

Windows V systému Windowsjsou všechny proměnné prostředí nastaveny automaticky během instalace.

Linux AIX V systému AIX and Linuxmůžete k nastavení proměnných prostředí použít skript `setjmsenv` (pokud používáte 32bitové prostředí JVM) nebo `setjmsenv64` (pokud používáte 64bitové prostředí JVM). Tyto skripty jsou v adresáři *MQ_INSTALLATION_PATH*/java/bin .

IBM i V systému IBM imusí být proměnná prostředí **QIBM_MULTI_THREADED** nastavena na Y. Poté můžete spustit aplikace s podporou podprocesů stejným způsobem, jakým spouštíte aplikace s podporou podprocesů. Další informace viz Nastavení IBM MQ pomocí Java a JMS.

IBM MQ classes for Java vyžadují prostředí Java 7 Java Runtime Environment (JRE). Informace o umístění vhodného prostředí JRE, které je nainstalováno s produktem IBM MQ, viz “Instalační adresáře pro IBM MQ classes for Java” na stránce 343.

IBM MQ classes for Java knihovny
Umístění knihoven IBM MQ classes for Java se liší v závislosti na platformě. Toto umístění zadejte při spuštění aplikace.

Chcete-li určit umístění knihoven Java Native Interface (JNI), spusťte aplikaci pomocí příkazu **java** v následujícím formátu:

```
java -Djava.library.path= library_path application_name
```

kde *cesta_k_knihovně* je cesta k souboru IBM MQ classes for Java, který zahrnuje knihovny JNI. Tabulka 53 na stránce 346 zobrazuje umístění knihoven IBM MQ classes for Java pro každou platformu. V této tabulce představuje *MQ_INSTALLATION_PATH* adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Tabulka 53. Umístění knihoven 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)
Linux (x86)	*MQ_INSTALLATION_PATH*/java/lib
Linux (platformy POWER, x86-64 a zSeries s390x)	*MQ_INSTALLATION_PATH*/java/lib (32bitové knihovny) *MQ_INSTALLATION_PATH*/java/lib64 (64bitové knihovny)
Windows	*MQ_INSTALLATION_PATH*\Java\lib (32bitové knihovny) *MQ_INSTALLATION_PATH*\Java\lib64 (64bitové knihovny)
z/OS	*MQ_INSTALLATION_PATH*/mqm/V8R0M0/java/lib (32bitové a 64bitové knihovny)

Poznámka:

1. Linux AIX V systémech AIX nebo Linux (platforma Power) použijte 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 32bitové knihovny.
2. Windows V systému Windowsmůžete použít proměnnou prostředí PATH k určení umístění knihoven IBM MQ classes for Java namísto jejich umístění v příkazu **java** .
3. IBM i Chcete-li použít IBM MQ classes for Java v režimu vazeb na systému IBM i, ujistěte se, že knihovna QMQMJAVA je ve vašem seznamu knihoven.
4. z/OS V systému z/OSmůžete použít buď 32bitové, nebo 64bitové prostředí JVM (Java Virtual Machine). Nemusíte určovat, které knihovny se mají použít; produkt IBM MQ classes for Java může sám určit, které knihovny JNI se mají načíst.

Související pojmy

Použití produktu IBM MQ classes for Java
Po instalaci produktu IBM MQ classes for Javamůžete nakonfigurovat instalaci tak, abyste spustili vlastní aplikace.

Podpora pro OSGi s IBM MQ classes for Java
OSGi poskytuje rámec, který podporuje implementaci aplikací jako svazků balíků. Jako součást produktu IBM MQ classes for Javajsou dodávány tři svazky balíků OSGi.

OSGi poskytuje obecný, zabezpečený a spravovaný rámec Java , který podporuje implementaci aplikací, které přicházejí 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 dynamickým a rozšiřitelným způsobem.

IBM MQ classes for Java zahrnuje následující svazky balíků OSGi.

com.ibm.mq.osgi.java_*version_number*.jar
Soubory JAR, které umožňují aplikacím používat soubor IBM MQ classes for Java.

com.ibm.mq.jakarta.osgi.allclient_*version_number*.jar
V 9.3.0 JM 3.0 V 9.3.0 V případě systému Jakarta Messaging 3.0tento soubor JAR umožňuje aplikacím používat soubory IBM MQ classes for JMS i IBM MQ classes for Javaa také zahrnuje kód pro zpracování zpráv PCF.

com.ibm.mq.osgi.allclient_*version_number*.jar
JMS 2.0 V případě systému JMS 2.0tento soubor JAR umožňuje aplikacím používat soubory IBM MQ classes for JMS i IBM MQ classes for Javaa také zahrnuje kód pro zpracování zpráv PCF.

com.ibm.mq.jakarta.osgi.allclientprereqs_*version_number*.jar
V 9.3.0 JM 3.0 V 9.3.0 Pro systém Jakarta Messaging 3.0tento soubor JAR poskytuje nezbytné předpoklady pro `com.ibm.mq.jakarta.osgi.allclient_`*`version_number`*`.jar`.

com.ibm.mq.osgi.allclientprereqs_*version_number*.jar
JMS 2.0 Pro systém JMS 2.0tento soubor JAR poskytuje nezbytné předpoklady pro `com.ibm.mq.osgi.allclient_`*`version_number`*`.jar`.

kde *`version_number`* je číslo verze produktu IBM MQ , který je nainstalován.

Balíky jsou nainstalovány do podadresáře `java/lib/OSGi` vaší instalace produktu IBM MQ nebo do 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. Použití těchto svazků balíků odebere omezení, že nelze spustit jak IBM MQ classes for JMS , tak IBM MQ classes for Java v rámci stejného rámce OSGi. Všechna ostatní omezení však stále platí. Pro verze produktu IBM MQ před IBM MQ 8.0platí omezení použití volby IBM MQ classes for JMS nebo IBM MQ classes for Java .

Devět dalších balíků je také instalováno do podadresáře `java/lib/OSGi` vaší instalace produktu IBM MQ nebo do složky `java\lib\OSGi` na systému Windows. Tyto svazky balíků jsou součástí IBM MQ classes for JMSa nesmí být načteny do běhového prostředí OSGi, které má načtený svazek balíků IBM MQ classes for Java . Pokud je balík OSGi IBM MQ classes for Java načten do běhového prostředí OSGi, které má také načtené balíky IBM MQ classes for JMS , dojde k chybám, jak ukazuje následující příklad, když jsou spuštěny aplikace používající buď balík IBM MQ classes for Java , nebo balíky IBM MQ classes for JMS :

```
java.lang.ClassCastException: com.ibm.mq.MQException incompatible with com.ibm.mq.MQException
```

Svazek balíků OSGi pro IBM MQ classes for Java byl zapsán do specifikace OSGi Release 4; nefunguje v prostředí OSGi Release 3.

Musíte správně nastavit cestu k systému nebo cestu ke knihovně, aby běhové prostředí OSGi mohlo najít všechny požadované soubory DLL nebo sdílené knihovny.

Používáte-li svazek balíků OSGi pro IBM MQ classes for Java, nejsou třídy uživatelských procedur kanálu zapsané v produktu Java podporovány kvůli vnitřnímu problému při načítání tříd v prostředí s více zavaděči tříd, jako např. OSGi. Uživatelský balík si může být vědom balíku IBM MQ classes for Java , ale balík IBM MQ classes for Java si není vědom žádného uživatelského balíku. V důsledku toho nemůže zavaděč tříd použitý v balíku IBM MQ classes for Java načíst třídu uživatelské procedury kanálu, která je v uživatelském balíku.

Další informace o OSGi naleznete na webu OSGi alliance .

z/OS *Instalace systému IBM MQ classes for Java na systému z/OS*
V systému z/OSmusí knihovna STEPLIB použitá za běhu obsahovat knihovny IBM MQ SCSQAUTH a SCSQANLE.

V produktu z/OS UNIX System Servicesmůžete přidat tyto knihovny pomocí řádku ve svém produktu `.profile` , jak ukazuje následující příklad, a nahradit `thlqual` kvalifikátorem datové sady vysoké úrovně, který jste vybrali při instalaci produktu IBM MQ:

```
export STEPLIB=thlqual.SCSQAUTH:thlqual.SCSQANLE:$STEPLIB
```

V jiných prostředích obvykle potřebujete upravit spouštěcí JCL tak, aby zahrnoval 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 uvádí vlastnosti, které se používají ke konfiguraci IBM MQ classes for Java.

Formát konfiguračního souboru IBM MQ classes for Java je formát standardního souboru vlastností Java .

Ukázkový konfigurační soubor `mqjava.config`se dodává v podadresáři `bin` instalačního adresáře IBM MQ classes for Java . Tento soubor dokumentuje všechny podporované vlastnosti a jejich výchozí hodnoty.

Poznámka: Ukázkový konfigurační soubor se přepíše, když se instalace produktu IBM MQ upgraduje na budoucí opravnou sadu. Proto se doporučuje vytvořit kopii ukázkového konfiguračního souboru pro použití s aplikacemi.

Můžete zvolit název a umístění konfiguračního souboru IBM MQ classes for Java . Při spouštění aplikace použijte příkaz **java** v následujícím formátu:

```
java -Dcom.ibm.msg.client.config.location=config_file_url application_name
```

V příkazu *config_file_url* je jednotný lokátor prostředků (URL), který určuje název a umístění konfiguračního souboru IBM MQ classes for Java . Jsou podporovány 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` na lokálním systému Windows .

Při spuštění 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 konfigurační soubor nelze nalézt, produkt IBM MQ classes for Java použije výchozí hodnoty pro všechny vlastnosti. V případě potřeby můžete přepsat libovolnou vlastnost v konfiguračním souboru tak, že ji zadáte jako systémovou vlastnost v příkazu **java** .

Konfigurační soubor IBM MQ classes for Java lze použít s jakýmkoli podporovaným přenosem mezi aplikací a správcem front nebo zprostředkovatelem.

Potlačení vlastností uvedených 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 určené v konfiguračním souboru IBM MQ MQI client se však použijí pouze v případě, že se aplikace připojí ke správci front v režimu klienta.

V případě potřeby můžete přepsat libovolný atribut v konfiguračním souboru IBM MQ MQI client jeho uvedením jako vlastnosti v konfiguračním souboru IBM MQ classes for Java . Chcete-li přepsat atribut v konfiguračním souboru IBM MQ MQI client , použijte v konfiguračním souboru IBM MQ classes for Java položku s následujícím formátem:

```
com.ibm.mq.cfg.stanza.propName=propValue
```

Proměnné v položce mají následující význam:

sekce
Název sekce 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 uvedenou v konfiguračním souboru IBM MQ MQI client .

Případně můžete přepsat atribut v konfiguračním souboru IBM MQ MQI client zadáním vlastnosti jako systémové vlastnosti v příkazu **java** . K určení vlastnosti jako systémové vlastnosti použijte předchozí formát.

Pouze následující atributy v konfiguračním souboru IBM MQ MQI client jsou důležité pro IBM MQ classes for Java. Zadáte-li nebo potlačíte-li jiné atributy, nebude to mít žádný účinek. Všimněte si konkrétně, že `ChannelDefinitionFile` a `ChannelDefinitionDirectory` v sekci CHANNELS konfiguračního souboru klienta se nepoužívají. Podrobnosti o použití tabulky CCDT s IBM MQ classes for Javaviz “Použití tabulky definic kanálů klienta s volbou IBM MQ classes for Java” na stránce 364 .

Tabulka 54. Která sekce konfiguračního souboru klienta obsahuje který atribut

sekce	**Atribut**
CHANNELS stanza konfiguračního souboru klienta	Put1DefaultAlwaysSync
CHANNELS stanza konfiguračního souboru klienta	PasswordProtection
ClientExitSekce cesty konfiguračního souboru klienta	ExitsDefaultPath
ClientExitSekce cesty konfiguračního souboru klienta	ExitsDefaultPath64
ClientExitSekce cesty konfiguračního souboru klienta	JavaExitsCesta ke třídě
Sekce JMQI konfiguračního souboru klienta	useMQCSPauthentication
MessageBuffer sekce konfiguračního souboru klienta	MaximumSize
MessageBuffer sekce konfiguračního souboru klienta	PurgeTime
MessageBuffer sekce konfiguračního souboru klienta	UpdatePercentage
stanza TCP konfiguračního souboru klienta	ClntRcvBuffSize
stanza TCP konfiguračního souboru klienta	ClntSndBuffSize
stanza TCP konfiguračního souboru klienta	Vypršení časového limitu připojení
stanza TCP konfiguračního souboru klienta	KeepAlive

Další informace o konfiguraci produktu IBM MQ MQI client viz IBM MQ MQI client konfigurační soubor `mqclient.ini`.

Související úlohy

Trasování tříd IBM MQ pro aplikace v jazyce Java

Použití Java standardního trasování prostředí ke konfiguraci trasování Java
Použijte sekci Java Standardní nastavení trasování prostředí ke konfiguraci prostředku trasování IBM MQ classes for Java .

com.ibm.msg.client.commonservices.trace.outputName = *traceOutputNázev*
traceOutputName je název adresáře a souboru, do kterého se odesílá výstup trasování.

Standardně se informace o trasování zapisují do trasovacího souboru v aktuálním pracovním adresáři aplikace. Název trasovacího souboru závisí na prostředí, ve kterém je aplikace spuštěna:

- Z adresáře IBM MQ 9.1.5 a IBM MQ 9.1.0 Fix Pack 5:
  - Pokud aplikace načetla soubor IBM MQ classes for Java ze souboru JAR, který lze přemístit `com.ibm.mq.allclient.jar`, trasování se zapíše do souboru s názvem `mqjavaclient_%PID%.cl%u.trc`.
  - Pokud aplikace načetla soubor IBM MQ classes for Java ze souboru JAR `com.ibm.mq.jar`, trasování se zapíše do souboru s názvem `mqjava_%PID%.cl%u.trc`.
- Z adresáře IBM MQ 9.0.0 Fix Pack 2:
  - Pokud aplikace načetla soubor IBM MQ classes for Java ze souboru JAR, který lze přemístit `com.ibm.mq.allclient.jar`, trasování se zapíše do souboru s názvem `mqjavaclient_%PID%.trc`.
  - Pokud aplikace načetla soubor IBM MQ classes for Java ze souboru JAR `com.ibm.mq.jar`, trasování se zapíše do souboru s názvem `mqjava_%PID%.trc`.
- V případě systému IBM MQ classes for Java for IBM MQ 9.0.0 Fix Pack 1 nebo dřívějšího se trasování zapisuje do souboru s názvem `mqjms_%PID%.trc`.

kde *%PID%* je identifikátor procesu aplikace, která se trasuje, a *%u* je jedinečné číslo pro rozlišení souborů mezi podprocesy spouštějícími trasování pod různými zavaděči tříd Java .

Není-li ID procesu k dispozici, vygeneruje se náhodné číslo s předponou s písmenem `f`. Chcete-li zahrnout ID procesu do zadaného názvu souboru, použijte řetězec `%PID%`.

Zadáte-li alternativní adresář, musí existovat a musíte mít pro tento adresář oprávnění k zápisu. Pokud nemáte oprávnění k zápisu, výstup trasování se zapíše do souboru `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 zahrne 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 nevylučuje žádné balíky a třídy v produktu IBM MQ classes for JMS z trasování.

Poznámka: Můžete vyloučit balík, ale pak zahrnout podbalíky tohoto balíku. Pokud například vyloučíte balík `a.b` a zahrnete balík `a.b.x`, trasování zahrne 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 uvedena, na stejné úrovni, jak je zahrnutá, tak vyloučená.

com.ibm.msg.client.commonservices.trace.maxBytes = *maxArrayBajty*
maxArrayBytes je maximální počet bajtů, které jsou trasovány z libovolného bajtového pole.

Je-li parametr *maxArrayBytes* nastaven na kladné celé číslo, omezuje počet bajtů v bajtovém poli, které jsou zapsány do trasovacího souboru. Osekne bajtové pole po zapsání *maxArrayBytes* . Nastavení *maxArrayBytes* zmenšuje velikost výsledného trasovacího souboru a snižuje vliv trasování na výkon aplikace.

Hodnota `0` pro tuto vlastnost znamená, že do trasovacího souboru není odeslán žádný obsah žádného bajtového pole.

Výchozí hodnota je `-1`, která odebere jakýkoli limit počtu bajtů v bajtovém poli, které jsou odeslány do trasovacího souboru.

com.ibm.msg.client.commonservices.trace.limit = *maxTraceBajty*
maxTraceBytes je maximální počet bajtů zapsaných do výstupního souboru trasování.

maxTraceBytes pracuje s *traceCycles*. Pokud se počet zapsaných bajtů trasování blíží limitu, soubor se zavře a spustí se nový výstupní soubor trasování.

Hodnota `0` znamená, že výstupní soubor trasování má nulovou délku. Výchozí hodnota je `-1`, což znamená, že množství dat, která se mají zapsat do výstupního souboru trasování, je neomezené.

com.ibm.msg.client.commonservices.trace.count = *traceCycles*
traceCycles je počet výstupních souborů trasování, které se mají procházet.

Pokud aktuální výstupní soubor trasování dosáhne limitu uvedeného parametrem *maxTraceBytes*, soubor se zavře. Další trasovací výstup se zapíše do dalšího trasovacího výstupního souboru v pořadí. Každý výstupní soubor trasování je rozlišen číselnou příponou připojenou k názvu souboru. Aktuální nebo nejnovější výstupní soubor trasování je `mqjms.trc.0`, další nejnovější výstupní soubor trasování je `mqjms.trc.1`. Starší trasovací soubory následují stejný vzor číslování až do limitu.

Výchozí hodnota *traceCycles* je 1. Pokud je *traceCycles* 1, když aktuální výstupní soubor trasování dosáhne své maximální velikosti, soubor se zavře a odstraní. Spustí se 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*
traceParameters řídí, zda jsou parametry metody a návratové hodnoty zahrnuty do trasování.

traceParameters standardně zobrazuje TRUE. Je-li parametr *traceParameters* nastaven na hodnotu FALSE, budou trasovány pouze podpisy metod.

com.ibm.msg.client.commonservices.trace.startup = *spuštění*
Existuje inicializační fáze IBM MQ classes for Java , během které jsou prostředky přiděleny. Hlavní prostředek trasování je inicializován během fáze alokace prostředků.

Je-li parametr *startup* nastaven na hodnotu TRUE, použije se trasování spuštění. Informace o trasování jsou okamžitě vytvořeny a zahrnují nastavení všech komponent, včetně samotného prostředku trasování. Informace o trasování spuštění lze použít k diagnostice problémů s konfigurací. Informace o trasování spuštění se vždy zapisují do souboru `System.err`.

startup standardně zobrazuje FALSE.

Produkt *startup* je zkontrolován před dokončením inicializace. Z tohoto důvodu zadejte vlastnost na příkazovém řádku pouze jako systémovou vlastnost Java . Neuvádějte jej v konfiguračním souboru IBM MQ classes for Java .

com.ibm.msg.client.commonservices.trace.compress = *compressedTrace*
Nastavte *compressedTrace* na TRUE , chcete-li komprimovat výstup trasování.

Výchozí hodnota *compressedTrace* je FALSE.

Je-li parametr *compressedTrace* nastaven na hodnotu TRUE, výstup trasování je komprimován. Výchozí název výstupního souboru trasování má příponu `.trz`. Je-li komprese nastavena na hodnotu FALSE, což je výchozí hodnota, soubor má příponu `.trc` , která označuje, že je nekomprimovaný.

Avšak pokud byl název souboru pro výstup trasování uveden v souboru *traceOutputName* , použije se tento název; pro soubor se nepoužije žádná přípona.

Komprimovaný výstup trasování je menší než nekomprimovaný. Protože je méně I/O, lze jej zapsat rychleji než nekomprimované trasování. Komprimované trasování má menší vliv na výkon IBM MQ classes for Java než nekomprimované trasování.

Jsou-li nastaveny hodnoty *maxTraceBytes* a *traceCycles* , vytvoří se místo více nestrukturovaných souborů více komprimovaných trasovacích souborů.

Pokud příkaz IBM MQ classes for Java skončí neřízeným způsobem, komprimovaný trasovací soubor nemusí být platný. Z tohoto důvodu musí být komprese trasování použita pouze v případě, že se produkt IBM MQ classes for Java uzavře řízeným způsobem. Kompresi trasování používejte pouze v případě, že vyšetřované problémy nezpůsobí neočekávané zastavení prostředí JVM. Nepoužívejte kompresi trasování při diagnostice problémů, které mohou vést k `System.Halt()` vypnutí nebo nekontrolovanému nekontrolovanému ukončení prostředí JVM.

com.ibm.msg.client.commonservices.trace.level = *traceLevel*
traceLevel uvádí úroveň filtrování pro trasování. Definované úrovně trasování jsou 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, bude do trasování zapsán libovolný bod trasování s definovanou úrovní TRACE_EXCEPTION, TRACE_WARNINGnebo TRACE_INFO . Všechny ostatní trasovací body jsou vyloučeny.

com.ibm.msg.client.commonservices.trace.standalone = *standaloneTrace*

standaloneTrace řídí, zda se služba trasování klienta IBM MQ classes for Java používá v prostředí WebSphere Application Server .

Je-li parametr *standaloneTrace* nastaven na hodnotu TRUE, vlastnosti trasování klienta IBM MQ classes for Java se použijí k určení konfigurace trasování.

Pokud je 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 . Generované informace o trasování závisí na nastavení trasování aplikačního serveru.

Výchozí 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.

Mnoho velkých vývojových organizací používá tyto nástroje k centrální správě úložišť knihoven třetích stran.

Soubor IBM MQ classes for Java se skládá z několika souborů JAR. Při vývoji jazykových aplikací Java pomocí tohoto rozhraní API je na počítači, na kterém je aplikace vyvíjena, vyžadována instalace produktu IBM MQ Server, IBM MQ Client nebo IBM MQ Client SupportPac .

Chcete-li použít nástroj pro správu softwaru a přidat soubory JAR, které tvoří soubor IBM MQ classes for Java , do centrálně spravovaného úložiště, musí být dodrženy následující body:

- Úložiště nebo kontejner musí být k dispozici pouze vývojářům ve vaší organizaci. Distribuce mimo organizaci není povolena.

- Úložiště musí obsahovat úplnou a konzistentní sadu souborů JAR z jednoho vydání IBM MQ nebo opravné sady.
- Jste zodpovědní za aktualizaci úložiště s jakoukoli údržbou poskytovanou podporou IBM .

V produktu IBM MQ 8.0je třeba do úložiště nainstalovat soubor JAR `com.ibm.mq.allclient.jar` .

V systému IBM MQ 9.0jsou vyžadovány soubory JAR podpory poskytovatele zabezpečení Bouncy Castle a CMS . Další informace viz "IBM MQ classes for Java přemístitelné soubory JAR" na stránce 341 a Podpora pro jiná prostředí nežIBM JRE.

Nastavení po instalaci pro aplikace IBM MQ classes for Java

Po instalaci produktu IBM MQ classes for Javamůžete nakonfigurovat instalaci tak, abyste spustili vlastní aplikace.

Nezapomeňte zkontrolovat soubor README produktu IBM MQ , zda neobsahuje nejnovější informace nebo specifické informace o vašem prostředí. Nejnovější verze souboru README produktu je k dispozici na webové stránce IBM MQ, WebSphere MQ, a MQSeries product readmes .

Před pokusem o spuštění aplikace IBM MQ classes for Java v režimu vazeb se ujistěte, že jste nakonfigurovali IBM MQ , jak je popsáno v tématu Konfigurace.

Konfigurace správce front pro přijetí připojení klienta 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 serveru a spusťte program modulu listener.

Podrobnosti viz "Konfigurace správce front pro přijetí připojení klienta na platformě Multiplatforms" na stránce 1034.

Spuštění aplikací IBM MQ classes for Java pod Java security manager

IBM MQ classes for Java lze spustit s povoleným Java security manager . Chcete-li úspěšně spustit aplikace s povoleným produktem Java security manager , musíte nakonfigurovat prostředí Java Virtual Machine (JVM) s vhodným souborem definice zásady.

Nejjednodušším způsobem, jak vytvořit vhodný soubor definice zásady, je změnit soubor zásad dodávaný s prostředím Java runtime environment (JRE). Ve většině systémů je tento soubor uložen v adresáři `path lib/security/java.policy`vzhledem k vašemu adresáři JRE. Soubory zásad můžete upravit buď pomocí upřednostňovaného editoru, nebo pomocí programu policytool dodávaného s prostředím JRE.

Musíte udělit oprávnění k souboru `com.ibm.mq.jmqi.jar` , aby mohl:

- Vytvořit sokety (v režimu klienta)
- Načíst nativní knihovnu (v režimu vazeb)
- Číst různé vlastnosti z prostředí

Systémová vlastnost **`os.name`** musí být k dispozici pro IBM MQ classes for Java při spuštění pod Java security manager.

Pokud vaše aplikace Java používá produkt Java security manager, musíte přidat následující oprávnění do souboru `java.security.policy` , který používá aplikace, jinak budou do aplikace vyvolány výjimky:

```
permission java.lang.RuntimePermission "modifyThread";
```

Toto RuntimePermission je vyžadováno klientem v rámci správy přiřazení a uzavření multiplexních konverzací přes připojení TCP/IP ke správcům front.

Příklad záznamu souboru zásad

Zde je uveden příklad položky souboru zásad, která umožňuje úspěšné spuštění produktu IBM MQ classes for Java pod výchozím správcem zabezpečení. Nahraďte řetězec *MQ_INSTALLATION_PATH* v tomto příkladu umístěním, kde je v 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";

// Required for Java applications that use the Java Security Manager
permission java.lang.RuntimePermission "modifyThread";
};
```

Tento příklad souboru zásad umožňuje, aby IBM MQ classes for Java správně fungoval pod správcem zabezpečení, ale možná budete muset povolit, aby váš vlastní kód fungoval správně, dříve než budou vaše aplikace fungovat.

Ukázkový kód dodávaný s produktem IBM MQ classes for Java nebyl specificky povolen pro použití se správcem zabezpečení. Testy IVT jsou však spuštěny s tímto souborem zásad a výchozím správcem zabezpečení.

Důležité:

Prostředek trasování IBM MQ classes for Java vyžaduje další oprávnění, protože provádí další dotazy na systémové vlastnosti a také další operace systému souborů.

Vhodný soubor zásad zabezpečení šablony pro spuštění ve správci zabezpečení s povoleným trasováním je uveden v adresáři `samples/wmqjava` instalace produktu IBM MQ jako `example.security.policy`.

V případě výchozí instalace se soubor `example.security.policy` nachází na:

Windows
V `C:\Program Files\IBM\MQ\Tools\wmqjava\samples\example.security.policy`

Linux
V `/opt/mqm/samp/wmqjava/samples/example.security.policy`

Solaris
V `/opt/mqm/samp/wmqjava/samples/example.security.policy`

AIX
V `/usr/mqm/samp/wmqjava/samples/example.security.policy`

Spuštění aplikací IBM MQ classes for Java na serveru CICS Transaction Server
Aplikaci IBM MQ classes for Java lze spustit jako transakci na serveru CICS Transaction Server.

Chcete-li spustit aplikaci IBM MQ classes for Java jako transakci v části CICS Transakční server pro z/OS, postupujte takto:

1. Definujte aplikaci a transakci do produktu CICS pomocí dodané transakce CEDA.
2. Ujistěte se, že je adaptér IBM MQ CICS nainstalován ve vašem systému CICS . z/OS (Podrobnosti viz Použití IBM MQ s CICS .)
3. Ujistěte se, že prostředí JVM uvedené v souboru CICS obsahuje odpovídající položky CLASSPATH a LIBPATH.
4. Zahajte transakci pomocí kteréhokoli 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 MQIVP je dodáván s produktem IBM MQ classes for Java. Tento program můžete použít k testování všech režimů připojení IBM MQ classes for Java.

Program vyzve k zadání řady voleb a dalších dat, aby určil, který režim připojení chcete ověřit. K ověření instalace použijte následující postup:

1. Pokud se chystáte spustit program v režimu klienta, konfigurujte správce front podle popisu v části “Konfigurace správce front pro přijetí připojení klienta na platformě Multiplatforms” na stránce 1034. Fronta, která se má použít, je SYSTEM.DEFAULT.LOCAL.QUEUE
2. Pokud se chystáte spustit program v režimu klienta, viz také “Použití produktu IBM MQ classes for Java” na stránce 335.

   Proveďte zbývající kroky této procedury na systému, na kterém se chystáte spustit program.
3. Ujistěte se, že jste aktualizovali proměnnou prostředí CLASSPATH podle pokynů v části “Proměnné prostředí relevantní pro IBM MQ classes for Java” na stránce 344.

4. Změňte adresář na *MQ_INSTALLATION_PATH*/mqm/samp/wmqjava/samples, kde *MQ_INSTALLATION_PATH* je cesta k instalaci produktu IBM MQ . Pak na příkazovém řádku zadejte:

```
java -Djava.library.path= library_path MQIVP
```

kde *cesta_k_knihovně* je cesta ke knihovnám IBM MQ classes for Java (viz “IBM MQ classes for Java knihovny” na stránce 345).

Na výzvu označené (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 snaží:

- 1. Připojte se ke správci front
- 2. Otevřete frontu SYSTEM.DEFAULT.LOCAL.QUEUE, vložte zprávu do fronty, získejte zprávu z fronty a pak zavřete frontu
- 3. Odpojení od správce front
- 4. Vrátit zprávu, pokud jsou operace úspěšné

Zde je příklad výzev a odpovědí, které můžete vidět. Skutečné výzvy 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 náznaku označeném (1).
2. Zvolíte-li připojení k serveru, nezobrazí se výzvy označené jako (2).
3. IBM i V systému IBM imůžete zadat příkaz `java MQIVP` pouze z prostředí QShell. Případně můžete spustit aplikaci pomocí příkazu CL `RUNJVA CLASS(MQIVP)`.

Použití ukázkových aplikací IBM MQ classes for Java

Ukázkové aplikace IBM MQ classes for Java poskytují přehled obecných funkcí rozhraní API produktu IBM MQ classes for Java . Můžete je použít k ověření instalace a nastavení serveru systému zpráv a k usnadnění vytváření vlastních aplikací.

Informace o této úloze

Potřebujete-li pomoc při vytváření vlastních aplikací, můžete ukázkové aplikace použít jako výchozí bod. Pro každou aplikaci je k dispozici jak zdrojová, tak i zkompilovaná verze. Zkontrolujte ukázkový zdrojový kód a identifikujte klíčové kroky pro vytvoření jednotlivých požadovaných objektů pro vaši aplikaci (volbyMQQueueManager, MQConstants, MQMessage, MQPutMessagea MQDestination) a nastavte všechny specifické vlastnosti, které jsou zapotřebí k určení způsobu práce aplikace. Další informace viz téma “Psaní aplikací IBM MQ classes for Java” na stránce 360. Ukázky mohou být v budoucích verzích produktu IBM MQ Javazměněny.

Tabulka 55 na stránce 357 ukazuje, kde jsou na každé platformě nainstalovány ukázkové aplikace IBM MQ classes for Java :

Tabulka 55. Instalační adresáře pro ukázkové aplikace IBM MQ classes for Java	
Platforma	**Adresář**
AIX Linux	*MQ_INSTALLATION_PATH*`/samp/wmqjava/samples`
Windows	*MQ_INSTALLATION_PATH*`\tools\wmqjava\samples`
IBM i	`/qibm/proddata/mqm/java/samples/wmqjava/samples`
z/OS	*MQ_INSTALLATION_PATH*`/java/samples/wmqjava`

Tabulka 56 na stránce 357 zobrazuje sady ukázkových aplikací dodávaných s produktem IBM MQ classes for Java.

Tabulka 56. IBM MQ classes for Java ukázkové aplikace	
Název vzorku	**Popis**
IMSBridgeSample.java	Jednoduchý program k předvedení pomocí IMS Bridge s IBM MQ classes for Java.
MQIVP.java	Program pro ověření instalace produktu IBM MQ Java .
MQMessagePropertiesSample.java	Demonstruje použití rozhraní API vlastností zprávy.
MQPubSubApiSample.java	Demonstruje použití rozhraní API publikování/odběru.
MQSample.java	Jednoduchý program, který demonstruje vložení a získání zprávy z fronty.
MQSampleMessageManager.java	Obslužná třída pro zpracování zpráv v IBM MQ základních Java ukázkách.
mqjcivp.properties	Tento balík prostředků obsahuje zprávy používané programem pro ověření instalace produktu IBM MQ classes for Java (`MQIVP.java`).

IBM MQ classes for Java poskytuje skript s názvem `runjms` , který lze použít ke spuštění ukázkových aplikací. Tento skript nastaví prostředí IBM MQ tak, aby vám umožňovalo spouštět ukázkové aplikace IBM MQ classes for Java .

Tabulka 57 na stránce 357 zobrazuje umístění skriptu na každé platformě:

Tabulka 57. Umístění skriptu `runjms`	
Platforma	**Adresář**
AIX Linux	*MQ_INSTALLATION_PATH*`/java/bin/runjms`
Windows	*MQ_INSTALLATION_PATH*`\java\bin\runjms.bat`

Tabulka 57. Umístění skriptu `runjms` (pokračování)	
Platforma	**Adresář**

IBM i IBM i	`/qibm/proddata/mqm/java/bin/runjms`
	, nebo
	`/qibm/proddata/mqm/java/bin/runjms64`

z/OS z/OS	*MQ_INSTALLATION_PATH*`java/bin/runjms`

Chcete-li použít skript `runjms` k vyvolání ukázkové aplikace, postupujte takto:

Postup

1. Otevřete 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. Zadáním následujícího příkazu spusťte ukázku s těmito parametry:

   ```
   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 instalováno pro produkt IBM MQ classes for JMS” na stránce 85
Při instalaci produktu IBM MQ classes for JMSse vytvoří několik souborů a adresářů. V systému Windowsse některá konfigurace provádí během instalace automatickým nastavením proměnných prostředí. Na jiných platformách a v určitých prostředích Windows musíte před spuštěním aplikací IBM MQ classes for JMS nastavit proměnné prostředí.

Řešení problémů IBM MQ classes for Java

Na začátku spusťte ověřovací program instalace. Možná budete muset také použít prostředek trasování.

Pokud se aplikace nedokončí úspěšně, spusťte ověřovací program instalace a postupujte podle pokynů uvedených v diagnostických zprávách. Program pro ověření instalace je popsán v části “Ověření instalace produktu IBM MQ classes for Java” na stránce 355.

Pokud problémy pokračují a potřebujete kontaktovat servisní tým IBM , můžete být požádáni o zapnutí trasovacího zařízení. Proveďte to, jak je uvedeno v následujícím příkladu.

Chcete-li trasovat program `MQIVP` , postupujte takto:

- 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_k_knihovně* je cesta ke knihovnám IBM MQ classes for Java (viz “IBM MQ classes for Java knihovny” na stránce 345).

Další informace o použití trasování naleznete v tématu Trasování IBM MQ classes for Java aplikací.

z/OS MQ Adv. VUE Java připojitelnost klienta k dávkovým aplikacím spuštěným na z/OS

Za určitých podmínek se aplikace IBM MQ classes for Java v systému z/OS může připojit ke správci front v systému z/OS pomocí připojení klienta. Použití připojení klienta může zjednodušit topologie IBM MQ .

Pokud je aplikace IBM MQ classes for Java spuštěna v dávkovém prostředí pomocí připojení klienta a platí jedna z následujících podmínek, může se aplikace připojit ke vzdálenému správci front z/OS :

- LTS V 9.3.4 Kód IBM MQ classes for Java je IBM MQ 9.3.4 nebo novější, nebo Long Term Support s použitou opravou APAR PH56722 . Správce front může mít libovolnou podporovanou verzi.
- Připojovaný správce front je spuštěn s oprávněním IBM MQ Advanced for z/OS Value Unit Edition , a proto má parametr **ADVCAP** nastaven na hodnotu ENABLED. Správce front může mít libovolnou podporovanou verzi.

Další informace o IBM MQ Advanced for z/OS Value Unit Edition viz IBM MQ identifikátory produktů a informace o exportu.

See ZOBRAZENÍ SPRÁVCE FRONT for more information on **ADVCAP** and Začátek QMGR for more information on **QMGRPROD**.

Aplikace IBM MQ classes for Java v systému z/OS nemůže používat připojení v režimu klienta pro připojení ke správci front, který není spuštěn v systému z/OS

Pokud se aplikace IBM MQ classes for Java v systému z/OS pokusí připojit pomocí režimu klienta a není to povoleno, vrátí se hodnota MQRC_ENVIRONMENT_ERROR .

Podpora Advanced Message Security (AMS)

Aplikace klienta IBM MQ classes for Java mohou používat produkt AMS při připojování ke vzdáleným správcům front z/OS v souladu s podmínkami dříve popsanými v tomto tématu.

Chcete-li použít produkt AMS tímto způsobem, musí aplikace klienta použít typ úložiště klíčů `jceracfks` v adresáři `keystore.conf`, kde:

- Předpona názvu vlastnosti je `jceracfks` a tato předpona názvu nerozlišuje velká a malá písmena.
- Úložiště klíčů je svazek klíčů RACF .
- Hesla nejsou vyžadována a budou ignorována. Důvodem je, že svazky klíčů RACF nepoužívají hesla.
- Pokud zadáte poskytovatele, poskytovatel musí být `IBMJCE`.

Používáte-li produkt `jceracfks` s produktem AMS, musí být úložiště klíčů ve formátu: `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 rozlišuje velikost písmen.

Následující příklad používá standardní AMS svazek klíčů pro uživatele `JOHNDOE`:

```
jceracfks.keystore=safkeyring://JOHNDOE/drq.ams.keyring
```

Související pojmy

“JMS/Jakarta Messaging připojitelnost klienta k dávkovým aplikacím spuštěným na z/OS” na stránce 121

Za určitých podmínek se aplikace IBM MQ classes for JMS/Jakarta Messaging v systému z/OS může připojit ke správci front v systému z/OS pomocí připojení klienta. Použití připojení klienta může zjednodušit topologie IBM MQ .

Psaní aplikací IBM MQ classes for Java

Tato kolekce témat poskytuje informace, které vám pomohou při psaní aplikací Java pro interakci se systémy IBM MQ .

Chcete-li použít IBM MQ classes for Java pro přístup k frontám IBM MQ , zapisujete aplikace Java , které obsahují volání, do kterých se vkládají zprávy, a získáváte zprávy z front IBM MQ . Podrobnosti o jednotlivých třídách viz 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 rozhraní API IBM MQ používá slovesa, která působí na objekty. Programovací rozhraní Java používá objekty, které se používají při volání metod.

Procedurální programovací rozhraní aplikace IBM MQ je sestaveno na sloveso, jako například:

```
MQBACK, MQBEGIN, MQCLOSE, MQCONN, MQDISC,
MQGET, MQINQ, MQOPEN, MQPUT, MQSET, MQSUB
```

Všechna tato příkazová slova přijímají jako parametr popisovač objektu IBM MQ , se kterým mají pracovat. Váš program se skládá ze sady objektů IBM MQ , které se používají při volání metod pro tyto objekty.

Používáte-li procedurální rozhraní, odpojíte se od správce front pomocí volání MQDISC (Hconn, CompCode, Reason), kde *Hconn* je manipulátor pro správce front.

V rozhraní Java je správce front reprezentován objektem třídy MQQueueManager. Od správce front se odpojíte voláním metody disconnect () pro tuto třídu.

```
// declare an object of type queue manager
MQQueueManager queueManager=new MQQueueManager();
...
// do something...
...
// disconnect from the queue manager
queueManager.disconnect();
```

IBM MQ classes for Java režimy připojení

Způsob, jakým programujete pro produkt IBM MQ classes for Java , závisí na režimech připojení, které chcete použít.

Pokud používáte připojení klienta, existuje řada rozdílů od produktu IBM MQ MQI client , ale je koncepčně podobný. Pokud používáte režim vazeb, můžete použít vazby rychlé cesty a můžete zadat příkaz MQBEGIN. Režim, který se má použít, určíte nastavením proměnných ve třídě MQEnvironment.

IBM MQ classes for Java připojení klientů

Když se IBM MQ classes for Java používá jako klient, je to jako IBM MQ MQI client, ale má řadu rozdílů.

Pokud programujete produkt *IBM MQ classes for Java* pro použití jako klient, mějte na paměti následující rozdíly:

- Podporuje pouze protokol TCP/IP.
- Při spuštění nečte žádné proměnné prostředí IBM MQ .
- Informace, které by byly uloženy v definici kanálu a v proměnných prostředí, mohou být uloženy ve třídě nazvané Prostředí. Alternativně mohou být tyto informace předány jako parametry při vytvoření připojení.

- Podmínky chyb a výjimek jsou zapsány do protokolu uvedeného ve třídě `MQException` . Výchozí cíl chyby je konzola Java .
- Pouze následující atributy v konfiguračním souboru klienta IBM MQ jsou důležité pro IBM MQ classes for Java. Zadáte-li jiné atributy, budou neúčinné.

sekce	**Atribut**
ClientExitSekce cesty konfiguračního souboru klienta	ExitsDefaultPath
ClientExitSekce cesty konfiguračního souboru klienta	ExitsDefaultPath64
ClientExitSekce cesty konfiguračního souboru klienta	JavaExitsClasspath
MessageBuffer sekce konfiguračního souboru klienta	MaximumSize
MessageBuffer sekce konfiguračního souboru klienta	PurgeTime
MessageBuffer sekce konfiguračního souboru klienta	UpdatePercentage
stanza TCP konfiguračního souboru klienta	ClntRcvBuffSize
stanza TCP konfiguračního souboru klienta	ClntSndBuffSize
stanza TCP konfiguračního souboru klienta	Vypršení časového limitu připojení
stanza TCP konfiguračního souboru klienta	KeepAlive

- Pokud se připojujete ke správci front, který vyžaduje převod znakových dat, může nyní klient V7 Java provést převod, pokud to správce front nemůže provést. Prostředí JVM klienta musí podporovat převod mezi CCSID klienta a CCSID správce front.
- Prostředí IBM MQ classes for Java nepodporuje automatické opětovné připojování klientů.

Při použití v režimu klienta produkt *IBM MQ classes for Java* nepodporuje volání MQBEGIN.

IBM MQ classes for Java režim vazeb
Režim vazeb IBM MQ classes for Java se liší od režimu klienta třemi hlavními způsoby.

Při použití v režimu vazeb produkt IBM MQ classes for Java používá rozhraní `JNI` (Java Native Interface) k přímému volání do existujícího rozhraní API správce front, nikoli ke komunikaci prostřednictvím sítě.

Standardně se aplikace, které používají IBM MQ classes for Java v režimu vazeb, připojují ke správci front pomocí volby *ConnectOption*, MQCNO_STANDARD_BINDINGS.

IBM MQ classes for Java podporuje následující *ConnectOptions*:

- MQCNO_FASTPATH_BINDING
- MQCNO_STANDARDNÍ_VAZBA
- MQCNO_SHARED_BINDING
- MQCNO_ISOLATED_BINDING

Další informace o *ConnectOptions*viz “Připojení ke správci front pomocí volání MQCONNX” na stránce 715.

Režim vazeb podporuje volání MQBEGIN za účelem zahájení globálních jednotek práce koordinovaných správcem front na všech platformách kromě platforem IBM MQ for IBM i a IBM MQ for z/OS.

Většina parametrů poskytovaných třídou `MQEnvironment` není relevantní pro režim vazeb a jsou ignorovány.

Definování, které připojení IBM MQ classes for Java se má použít
Typ připojení, které se má použít, je určen nastavením proměnných ve třídě MQEnvironment.

Používají se 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 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:

- Pro připojení klienta 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 na správcích front
Tato kolekce témat popisuje, jak se připojit ke správci front a jak se od něj odpojit pomocí produktu IBM MQ classes for Java.

Nastavení prostředí IBM MQ pro IBM MQ classes for Java
Aby se aplikace mohla připojit ke správci front v režimu klienta, musí zadat název kanálu, název hostitele a číslo portu.

Poznámka: Informace v tomto tématu jsou relevantní pouze v případě, že se aplikace připojuje ke správci front v režimu klienta. Není relevantní, pokud se připojuje v režimu vazeb. Viz “Režimy připojení pro IBM MQ classes for JMS” na stránce 105.

Název kanálu, název hostitele a číslo portu můžete zadat jedním ze dvou způsobů: buď jako pole ve třídě MQEnvironment, nebo jako vlastnosti objektu MQQueueManager .

Pokud nastavíte pole ve třídě MQEnvironment, použijí se pro celou aplikaci, s výjimkou případů, kdy jsou přepsána hašovací tabulkou vlastností. Chcete-li zadat název kanálu a název hostitele v prostředí MQEnvironment, použijte následující kód:

```
MQEnvironment.hostname = "host.domain.com";
MQEnvironment.channel = "java.client.channel";
```

Jedná se o ekvivalent k nastavení proměnné prostředí **MQSERVER** :

```
"java.client.channel/TCP/host.domain.com".
```

Standardně se klienti Java pokusí připojit k modulu listener IBM MQ na portu 1414. Chcete-li zadat jiný port, použijte následující kód:

```
MQEnvironment.port = nnnn;
```

kde nnnn je požadované číslo portu

Předáte-li vlastnosti objektu správce front při jeho vytvoření, platí pouze pro tohoto správce front. Vytvořte položky v objektu hašovací tabulky s klíči **hostname**, **channel**a volitelně **port**a s odpovídajícími 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 je zobrazen příkazem **DISPLAY CONN MQSC/PCF** (kde se pole nazývá **APPLTAG**). nebo v zobrazení IBM MQ Průzkumník **Připojení aplikací** (kde se pole nazývá **App name**).

Názvy aplikací jsou omezeny na 28 znaků, takže delší názvy jsou oříznuty. Není-li uveden název aplikace, je poskytnuta výchozí hodnota. Výchozí název je založen na vyvolávající (hlavní) třídě, ale pokud tyto informace nejsou k dispozici, použije se text `IBM MQ Klient pro jazyk Java` .

Je-li použit název vyvolávající třídy, je přizpůsoben tak, aby se vešel do názvu odebráním úvodních názvů balíků, je-li to nutné. Pokud je například vyvolávající třída `com.example.MainApp`, použije se úplný název, ale pokud je vyvolávající třída `com.example.dictionaryAndThesaurus.multilingual.mainApp`, použije se název `multilingual.mainApp` , protože se jedná o nejdelší kombinaci názvu třídy a nejpravějšího názvu balíku, který odpovídá dostupné délce.

Je-li samotný název třídy delší než 28 znaků, je oříznut tak, aby se vešel. Například `com.example.mainApplicationForSecondTestCase` se změní na `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 je předána konstruktoru MQQueueManager , přidejte název do hašovací tabulky vlastností s klíčem **MQConstants.APPNAME_PROPERTY**.

Potlačení vlastností uvedených 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 použijí pouze v případě, že se aplikace připojí ke správci front v režimu klienta.

V případě potřeby můžete přepsat libovolný atribut v konfiguračním souboru IBM MQ jedním z následujících způsobů. Volby jsou zobrazeny v pořadí podle priority.

- Nastavte systémovou vlastnost Java pro vlastnost konfigurace.
- Nastavte vlastnost v mapě MQEnvironment.properties .
- V systému Java5 a novějších vydáních nastavte systémovou proměnnou prostředí.

Pouze následující atributy v konfiguračním souboru klienta IBM MQ jsou důležité pro IBM MQ classes for Java. Zadáte-li nebo potlačíte-li jiné atributy, nebude to mít žádný účinek.

sekce	Atribut
ClientExitSekce cesty konfiguračního souboru klienta	ExitsDefaultPath
ClientExitSekce cesty konfiguračního souboru klienta	ExitsDefaultPath64
ClientExitSekce cesty konfiguračního souboru klienta	JavaExitsClasspath
MessageBuffer sekce konfiguračního souboru klienta	MaximumSize
MessageBuffer sekce konfiguračního souboru klienta	PurgeTime

sekce	Atribut
MessageBuffer sekce konfiguračního souboru klienta	UpdatePercentage
stanza TCP konfiguračního souboru klienta	ClntRcvBufSize
stanza TCP konfiguračního souboru klienta	ClntSndBufSize
stanza TCP konfiguračního souboru klienta	Vypršení časového limitu připojení
stanza TCP konfiguračního souboru klienta	KeepAlive

Připojení ke správci front v adresáři IBM MQ classes for Java
Připojte se ke správci front vytvořením nové instance třídy MQQueueManager . Odpojte se 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, zavolejte metodu disconnect () ve správci front:

```
queueManager.disconnect();
```

Pokud zavoláte metodu odpojení, všechny otevřené fronty a procesy, ke kterým jste přistupovali prostřednictvím tohoto správce front, se zavřou. Je však dobrým zvykem tyto prostředky explicitně zavřít, až je dokončíte používat. K tomu použijte metodu close () na příslušných objektech.

Metody commit () a backout () ve správci front jsou ekvivalentní voláním MQCMIT a MQBACK používaným s procedurálním rozhraním.

Použití tabulky definic kanálů klienta s volbou IBM MQ classes for Java
Klientská aplikace IBM MQ classes for Java může používat definice kanálů připojení klienta uložené v tabulce CCDT (Client Channel Definition Table).

Jako alternativu k vytvoření definice kanálu připojení klienta nastavením určitých polí a vlastností prostředí ve třídě `MQEnvironment` nebo jejich předáním do souboru `MQQueueManager` v hašovací tabulce vlastností může aplikace klienta IBM MQ classes for Java používat definice kanálu připojení klienta, které jsou uloženy v tabulce definic kanálu klienta. Tyto definice jsou vytvořeny příkazy jazyka MQSC (IBM MQ Script) nebo PCF (IBM MQ Programmable Command Format) nebo pomocí konzoly IBM MQ Explorer .

Když aplikace vytvoří objekt `MQQueueManager` , klient IBM MQ classes for Java vyhledá v tabulce definic kanálů klienta vhodnou definici kanálu připojení klienta a použije tuto definici kanálu ke spuštění kanálu MQI. Další informace o definičních tabulkách kanálů klienta a o způsobu jejich konstrukce naleznete v tématu Tabulka definic kanálů klienta.

Chcete-li použít tabulku definic kanálů klienta, musí aplikace nejprve vytvořit objekt URL . Objekt URL zapouzdřuje jednotný lokátor prostředků (URL), který 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, ve kterém je aplikace spuštěna, může aplikace vytvořit objekt 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 systému, v němž je aplikace spuštěna. Pokud lze k souboru přistupovat pomocí protokolu FTP, může aplikace vytvořit objekt URL následujícím způsobem:

```
java.net.URL chanTab2 = new URL("ftp://ftp.server/admdata/ccdt2.tab");
```

Poté, co aplikace vytvoří objekt URL , může vytvořit objekt `MQQueueManager` pomocí jednoho z konstruktorů, které jako parametr převezme objekt URL . Příklad:

```
MQQueueManager mars = new MQQueueManager("MARS", chanTab2);
```

Tento příkaz způsobí, že klient IBM MQ classes for Java bude přistupovat k tabulce definic kanálů klienta určené objektem URL chanTab2, vyhledá v tabulce vhodnou definici kanálu připojení klienta a poté pomocí definice kanálu spustí kanál MQI pro správce front s názvem MARS.

Všimněte si následujících bodů, které platí v případě, že aplikace používá tabulku definic kanálů klienta:

- Když aplikace vytvoří objekt `MQQueueManager` pomocí konstruktoru, který jako parametr vezme objekt URL , nesmí být ve třídě `MQEnvironment` nastaven žádný název kanálu, buď jako pole, nebo jako vlastnost prostředí. Je-li nastaven název kanálu, klient IBM MQ classes for Java vygeneruje výjimku `MQException`. Pole nebo vlastnost prostředí určující název kanálu je považována za nastavenou, pokud má 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 stejné hodnoty, které lze použít pro parametr **QMgrName** ve volání MQCONN vydaném klientskou aplikací, která používá rozhraní MQI (Message Queue Interface). Další informace o významu těchto hodnot viz“Přehled rozhraní fronty zpráv” na stránce 700.

  Pokud vaše aplikace používá sdružování připojení, viz“Řízení výchozího fondu připojení v adresáři IBM MQ classes for Java” na stránce 384.
- Pokud klient IBM MQ classes for Java najde v tabulce definic kanálů klienta vhodnou definici kanálu připojení klienta, použije ke spuštění kanálu MQI pouze informace extrahované z této definice kanálu. Všechna pole související s kanálem nebo vlastnosti prostředí, které aplikace mohla nastavit ve třídě `MQEnvironment` , jsou ignorovány.

  Všimněte si zejména následujících bodů, pokud používáte TLS (Transport Layer Security):
  - Kanál MQI používá protokol TLS pouze v případě, že definice kanálu extrahovaná z tabulky definic kanálů klienta určuje název CipherSpec podporované klientem IBM MQ classes for Java .
  - Tabulka definic kanálů klienta obsahuje také informace o umístění serverů LDAP (Lightweight Directory Access Protocol), které obsahují seznamy odvolaných certifikátů (CRL). Klient IBM MQ classes for Java používá pouze tyto informace pro přístup k serverům LDAP, které obsahují seznamy CRL.
  - Tabulka definic kanálů 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. Protokol OCSP však můžete konfigurovat podle popisu v části Použití online protokolu certifikátů .

  Další informace o použití protokolu TLS s tabulkou definic kanálů klienta naleznete v tématu Určení, že kanál MQI používá protokol TLS.

  Všimněte si také následujících bodů, pokud používáte uživatelské procedury kanálu:

- Kanál MQI používá uživatelské procedury kanálu a přidružená uživatelská data určená definicí kanálu extrahovanou z tabulky definic kanálů klienta namísto uživatelských procedur kanálu a dat určených jinými metodami.
- Definice kanálu extrahovaná z tabulky definic kanálů klienta může určovat uživatelské procedury kanálu, které jsou napsány v adresáři Java, C nebo C + +. Další informace o zápisu uživatelské procedury kanálu v adresáři Java naleznete v části"Vytvoření uživatelské procedury kanálu v adresáři IBM MQ classes for Java" na stránce 378. Další informace o způsobu zápisu uživatelské procedury kanálu v jiných jazycích naleznete v části "Použití uživatelských procedur kanálu, které nejsou zapsány v souboru Java , s IBM MQ classes for Java" na stránce 381.

Určení rozsahu portů pro připojení klienta IBM MQ classes for Java
Můžete zadat port nebo rozsah portů, ke kterým se může aplikace vázat jedním ze dvou způsobů.

Když se aplikace IBM MQ classes for Java pokusí připojit ke správci front IBM MQ v režimu klienta, může brána firewall povolit pouze ta připojení, která pocházejí ze zadaných portů nebo rozsahu portů. V této situaci můžete uvést port nebo rozsah portů, ke kterým se může aplikace vázat. Porty můžete určit následujícími způsoby:

- Můžete nastavit pole nastavení localAddressve 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)");
```

- Při vytváření objektu MQQueueManager můžete předat hašovací tabulku vlastností obsahující vlastnost LOCAL_ADDRESS_PROPERTY s hodnotou "192.0.2.0(2000,3000)".

V každém z těchto příkladů platí, že když se aplikace později připojí ke správci front, aplikace se váže na lokální adresu IP a číslo portu v rozsahu 192.0.2.0(2000) až 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 pro připojení použito.

Pokud omezíte rozsah portů, mohou se vyskytnout chyby připojení. Dojde-li k chybě, dojde k výjimce MQException obsahující kód příčiny IBM MQ MQRC_Q_MGR_NOT_AVAILABLE a následující zprávu:

```
Socket connection attempt refused due to LOCAL_ADDRESS_PROPERTY restrictions
```

K chybě může dojít, pokud jsou všechny porty v uvedeném rozsahu používány, nebo pokud je zadaná adresa IP, název hostitele nebo číslo portu neplatné (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 . Struktura deskriptoru objektu MQOD (object descriptor structure) je sbalena do parametrů těchto metod.

Fronty

Chcete-li otevřít frontu, můžete použít metodu accessQueue třídy MQQueueManager . Například ve správci front s názvem queueManagerpoužijte následující kód:

```
MQQueue queue = queueManager.accessQueue("qName",CMQC.MQOO_OUTPUT);
```

Metoda accessQueue vrací nový objekt třídy MQQueue.

Po dokončení použití fronty ji zavřete pomocí metody close (), jak je uvedeno v následujícím příkladu:

```
queue.close();
```

Můžete také vytvořit frontu pomocí konstruktoru MQQueue. Parametry jsou přesně stejné jako pro metodu accessQueue s přidáním parametru správce front. Příklad:

```
MQQueue queue = new MQQueue(queueManager,
                            "qName",
                            CMQC.MQOO_OUTPUT,
                            "qMgrName",
                            "dynamicQName",
                            "altUserID");
```

Při vytváření front můžete zadat řadu voleb. Podrobnosti naleznete v tématu Class.com.ibm.mq.MQQueue. Vytvoření objektu fronty tímto způsobem umožňuje psát vlastní podtřídy MQQueue.

Témata

Podobně můžete téma otevřít pomocí metody accessTopic třídy MQQueueManager . Například ve správci front s názvem queueManagervytvořte odběratele a vydavatele pomocí následujícího kódu:

```
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 tématu jej zavřete pomocí metody close ().

Můžete také vytvořit téma pomocí konstruktoru MQTopic. Parametry jsou přesně stejné jako pro metodu accessTopic 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 naleznete v tématu Třída com.ibm.mq.MQTopic. Vytvoření objektu tématu tímto způsobem umožňuje psát vlastní podtřídy objektu 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 obojí). Ty lze použít pouze k otevření tématu pro odběry. Zbývající dvě metody mají parametr **openAs** a téma lze otevřít pro publikování nebo odběr v závislosti na hodnotě parametru **openAs** .

Chcete-li vytvořit téma jako trvalého odběratele, použijte buď metodu accessTopic třídy MQQueueManager , nebo konstruktor MQTopic, který přijímá název odběru, a v obou případech nastavte CMQC.MQSO_DURABLE .

Procesy

Chcete-li získat přístup k procesu, použijte metodu accessProcess správce 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 získat přístup k procesu, použijte metodu accessProcess správce MQQueueManager.

Metoda accessProcess vrací nový objekt třídy MQProcess.

Po dokončení použití objektu procesu jej zavřete pomocí metody close (), jak je uvedeno v následujícím příkladu:

```
process.close();
```

Můžete také vytvořit proces pomocí konstruktoru MQProcess. Parametry jsou přesně stejné jako pro metodu accessProcess s přidáním parametru správce front. Příklad:

```
MQProcess process =
      new MQProcess(queueManager,"PROCESSNAME",
      CMQC.MQOO_FAIL_IF_QUIESCING);
```

Vytvoření objektu procesu tímto způsobem umožňuje psát vlastní podtřídy MQProcess.

Zpracování zpráv v adresáři IBM MQ classes for Java

Zprávy jsou reprezentovány třídou MQMessage. Zprávy lze vkládat a získávat pomocí metod třídy MQDestination, která obsahuje podtřídy MQQueue a MQTopic.

Vkládání zpráv do front nebo témat pomocí metody put () třídy MQDestination. Zprávy jsou získávány z front nebo témat pomocí metody get () třídy MQDestination. Na rozdíl od procedurálního rozhraní, kde MQPUT a MQGET vkládají a dostávají pole bajtů, programovací jazyk Java vkládá a získává instance třídy MQMessage. Třída MQMessage zapouzdřuje datovou vyrovnávací paměť, která obsahuje skutečná data zprávy, spolu se všemi parametry MQMD (deskriptor zprávy) a vlastnostmi zprávy, které popisují tuto zprávu.

Chcete-li sestavit novou zprávu, vytvořte novou instanci třídy MQMessage a pomocí metod writeXXX vložte data do vyrovnávací paměti zpráv.

Při vytvoření nové instance zprávy jsou všechny parametry MQMD automaticky nastaveny na výchozí hodnoty, jak je definováno v poli Počáteční hodnoty a deklarace jazyka pro MQMD. Metoda put () objektu MQDestination také bere jako parametr instanci třídy MQPutMessageOptions. 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 () objektu MQDestination vrací novou instanci MQMessage, která představuje zprávu právě převzatou z fronty. Jako parametr bere také instanci třídy voleb MQGetMessage. Tato třída představuje strukturu MQGMO.

Není třeba zadávat maximální velikost zprávy, protože metoda get () automaticky upravuje velikost vnitřní vyrovnávací paměti tak, aby odpovídala příchozí zprávě. 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 čísel, který používají metody čtení a zápisu, můžete změnit nastavením členské proměnné *kódování* .

Znakovou sadu, která se má použít pro čtení a zápis řetězců, můžete změnit nastavením členské proměnné *characterSet* .

Další informace viz Třída MQMessage .

Poznámka: Metoda writeUTF() MQMessage automaticky kóduje délku řetězce, stejně jako bajty Unicode, které obsahuje. Když bude vaše zpráva přečtena jiným programem Java (pomocí readUTF()), jedná se o 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 spotřebovávání přechodných zpráv z klientské aplikace, můžete použít *dopředné čtení*. Klientské aplikace používající příkaz MQGET nebo asynchronní spotřebu budou mít prospěch ze zlepšení výkonu při procházení zpráv nebo spotřebovávání přechodných zpráv.

Obecné informace o zařízení pro dopředné čtení naleznete v souvisejícím tématu.

V produktu IBM MQ classes for Javapoužíváte produkt CMQC.MQSO_READ_AHEAD a CMQC.MQSO_NO_READ_AHEAD objektu MQQueue nebo MQTopic, které určují, zda mají spotřebitelé zpráv a prohlížeče front povoleno používat pro tento objekt dopředné čtení.

Asynchronní vkládání zpráv pomocí IBM MQ classes for Java
Chcete-li vložit zprávu asynchronně, nastavte parametr MQPMO_ASYNC_RESPONSE.

Zprávy vkládáte do front nebo témat pomocí metody put () třídy MQDestination. Chcete-li vložit zprávu asynchronně, tj. umožnit dokončení operace bez čekání na odpověď od správce front, můžete nastavit volbu MQPMO_ASYNC_RESPONSE v poli voleb MQPutMessage. Chcete-li určit úspěch nebo selhání asynchronních operací vložení, použijte stavové volání MQQueueManager.getAsync.

Publikovat/odebírat v produktu IBM MQ classes for Java
V produktu IBM MQ classes for Javaje téma reprezentováno třídou MQTopic a publikujete do něj pomocí metod MQTopic.put().

Obecné informace o publikování/odběru produktu IBM MQ naleznete v tématu Publikování/odběr systému zpráv.

Obsluha záhlaví zpráv IBM MQ pomocí IBM MQ classes for Java
Jsou poskytnuty třídy Java představující různé typy záhlaví zprávy. K dispozici jsou také dvě pomocné třídy.

Rozhraní MQHeader

Objekty záhlaví jsou popsány rozhraním MQHeader, které poskytuje obecné metody pro přístup k polím záhlaví a pro čtení a zápis obsahu zprávy. Každý typ záhlaví má svou vlastní třídu, která implementuje rozhraní MQHeader a přidává metody getter a setter pro jednotlivá pole. Typ záhlaví MQRFH2 je například reprezentován třídou MQRFH2 ; typ záhlaví MQDLH třídou MQDLH atd. Třídy záhlaví automaticky provádějí potřebnou konverzi dat a mohou číst nebo zapisovat data v libovolném uvedeném číselném kódování nebo znakové sadě (CCSID).

Důležité: Třídy záhlaví MQRFH2 považují zprávu za soubor s náhodným přístupem, což znamená, že kurzor musí být umístěn na začátku zprávy. Před použitím interní třídy záhlaví zprávy, například MQRFH,

MQRFH2, MQCIH, MQDEAD, MQIIH nebo MQXMIT, se ujistěte, že jste před předáním zprávy do třídy aktualizovali pozici kurzoru zprávy na správné umístění.

Pomocné třídy

Dvě pomocné třídy, MQHeaderIterator a MQHeaderList, pomáhají při čtení a dekódování (syntaktické analýze) obsahu záhlaví ve zprávách:

- Třída MQHeaderIterator funguje jako java.util.Iterator. Dokud je ve zprávě více záhlaví, vrátí metoda next () hodnotu true a metoda nextHeader() nebo next () vrátí další objekt záhlaví.
- MQHeaderList funguje jako java.util.List. Podobně jako MQHeaderIteratoranalyzuje obsah záhlaví, ale také vám umožňuje vyhledat konkrétní záhlaví, přidat nová záhlaví, odebrat existující záhlaví, aktualizovat pole záhlaví a poté zapsat obsah záhlaví zpět do zprávy. Případně můžete vytvořit prázdný seznam MQHeaderList, poté jej naplnit instancemi záhlaví a zapsat jej jednou nebo opakovaně do zprávy.

Třídy MQHeaderIterator a MQHeaderList používají informace v registru MQHeaderRegistry , aby poznali, které třídy záhlaví IBM MQ jsou přidruženy ke konkrétním typům a formátům zpráv. Registr MQHeaderRegistry je nakonfigurován se znalostí všech aktuálních formátů a typů záhlaví IBM MQ a jejich implementačních tříd a můžete také registrovat vlastní typy záhlaví.

Podpora je poskytována pro následující běžně používaná záhlaví IBM MQ

- MQRFH-Pravidla a formátování záhlaví
- MQRFH2 -Podobně jako MQRFH, používá se k předávání zpráv zprostředkovateli zpráv, který patří k produktu IBM Integration Bus, a z něj. Používá se také k uložení vlastností zprávy
- MQCIH- CICS most
- MQDLH-Záhlaví nedoručených zpráv
- Záhlaví informací MQIIH- IMS
- MQRMH-záhlaví referenční zprávy
- MQSAPH-záhlaví SAP
- MQWIH-záhlaví informací o práci
- MQXQH-Záhlaví přenosové fronty
- Záhlaví MQDH-Distribution
- MQEPH-Zapouzdřené záhlaví PCF

Můžete také definovat třídy představující vaše vlastní záhlaví.

Chcete-li použít MQHeaderIterator k získání záhlaví RFH2 , buď nastavte volbu MQGMO_PROPERTIES_FORCE_MQRFH2 ve volbách GetMessage, nebo nastavte vlastnost fronty PROPCTL FORCE.

Tisk všech záhlaví ve zprávě pomocí IBM MQ classes for Java
V tomto příkladu instance MQHeaderIterator analyzuje záhlaví v 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);
}
```

Přeskočení záhlaví ve zprávě pomocí IBM MQ classes for Java
V tomto příkladu umístí metoda skipHeaders() MQHeaderIterator kurzor čtení zprávy bezprostředně 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 nedoručenou zprávou pomocí IBM MQ classes for Java
V tomto příkladu metoda čtení naplní objekt MQDLH čtením ze zprávy. Po operaci čtení je kurzor čtení zprávy umístěn bezprostředně za obsahem záhlaví MQDLH.

Zprávy ve frontě nedoručených zpráv správce front mají předponu v záhlaví nedoručených zpráv (MQDLH). Chcete-li se rozhodnout, jak s těmito zprávami zacházet-například chcete-li určit, zda je zopakovat nebo vyřadit-, musí aplikace pracující s nedoručenou zprávou zkontrolovat kód příčiny obsažený v 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 pohodlí, který se sám inicializuje 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 nedoručených zpráv pomocí IBM MQ classes for Java
V tomto příkladu se MQDLH používá k odebrání záhlaví ze zprávy nedoručených zpráv.

Aplikace ošetřující nedoručené zprávy obvykle znovu odešle zprávy, které byly odmítnuty, pokud jejich kód příčiny označuje přechodnou chybu. Před opětovným odesláním zprávy musí odebrat záhlaví MQDLH.

Tento příklad provede následující kroky (viz komentáře v ukázkovém kódu):

1. Příkaz MQHeaderList přečte celou zprávu a každé záhlaví zjištěné ve zprávě se stane položkou v seznamu.
2. Zprávy nedoručených zpráv obsahují jako první záhlaví MQDLH, takže toto lze nalézt v první položce seznamu záhlaví. Objekt MQDLH byl již naplněn ze zprávy při sestavení seznamu MQHeaderList , takže není třeba vyvolávat jeho metodu čtení.
3. Kód příčiny je extrahován pomocí metody getReason() poskytnuté třídou MQDLH.
4. Kód příčiny byl zkontrolován a označuje, že je vhodné zprávu znovu odeslat. MQDLH je odebrán pomocí metody MQHeaderList remove ().
5. Příkaz MQHeaderList zapíše zbývající obsah do nového objektu zprávy. Nová zpráva nyní obsahuje vše v původní zprávě s výjimkou MQDLH a lze ji zapsat do fronty. Argument **true** pro konstruktor a metodu zápisu označuje, že tělo zprávy má být zadrženo v seznamu MQHeaderLista znovu zapsáno.

6. Pole formátu v deskriptoru nové zprávy nyní obsahuje hodnotu, která byla dříve v poli formátu MQDLH. Data zprávy odpovídají číselnému kódování a sadě CCSID v deskriptoru zprávy.

```
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 IBM MQ classes for Java
Tento příklad používá příkaz MQHeaderList k vytištění obsahu zprávy včetně jejích záhlaví.

Výstup obsahuje pohled na veškerý obsah záhlaví i tělo zprávy. Třída MQHeaderList dekóduje všechna záhlaví najednou, zatímco MQHeaderIterator je postupně prochází pod kontrolou aplikace. Tuto techniku můžete použít k poskytnutí jednoduchého ladicího nástroje při psaní aplikací 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 z polí deskriptoru zpráv zprávy zprávy MQMessage namísto čtení 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 IBM MQ classes for Java
Tento příklad používá metodu indexOf(String) objektu MQHeaderList k nalezení záhlaví MQRFH2 ve zprávě, pokud existuje.

```
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 IBM MQ classes for Java
Tento příklad ukazuje, jak přistupovat k 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 kódovaných pomocí XML, které jsou přenášeny v rámci pole NameValueData. Tento příklad ukazuje, jak přistupovat k známé hodnotě pole v pojmenované složce-v této instanci k poli Rto ve složce jms, která představuje název fronty odpovědí ve zprávě produktu MQ JMS .

```
MQRFH2 rfh = ...

String value = rfh.getStringFieldValue ("jms", "Rto");
```

Chcete-li zjistit obsah MQRFH2 (na rozdíl od přímého vyžádání specifických polí), můžete pomocí metody getFolders vrátit seznam MQRFH2.Element, který představuje strukturu složky, která může obsahovat pole a další složky. Nastavení pole nebo složky na hodnotu null jej odebere z MQRFH2. Při manipulaci s obsahem složky NameValueData tímto způsobem se pole StrucLength automaticky aktualizuje.

Čtení a zápis bajtových proudů jiných než objektů MQMessage pomocí IBM MQ classes for Java
Tyto příklady používají třídy záhlaví k analýze a manipulaci s obsahem záhlaví IBM MQ , když zdroj dat není objekt MQMessage.

Třídy záhlaví můžete použít k analýze a manipulaci s obsahem záhlaví IBM MQ , i když je zdroj dat něco jiného než objekt MQMessage. Rozhraní MQHeader implementované 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 pro čtení a zápis obsahu MQMessage a přepsat kódování a CCSID uvedené v deskriptoru zprávy. To je užitečné pro zprávy, které obsahují řetězec záhlaví v různých kódováních.

Můžete také získat objekty DataInput a DataOutput z jiných datových proudů, například proudů souborů nebo soketů, nebo bajtových polí přenášených ve zprávách JMS . Třídy java.io.DataInputStream implementují 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 začínající MQHeaderIterator lze nahradit

```
MQDLH dlh = new MQDLH (in, CMQC.MQENC_NATIVE, CMQC.MQCCSI_DEFAULT);
// or any other header type
```

Tento příklad zapisuje do bajtového 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 ();
```

Při práci s proudy tímto způsobem dbejte na to, abyste použili správné hodnoty pro kódování a argumenty characterSet . Při čtení záhlaví zadejte kódování a CCSID, se kterým byl bajtový obsah původně zapsán. Při zápisu záhlaví zadejte kódování a CCSID, které chcete vytvořit. Převod dat se provádí automaticky třídami záhlaví.

Vytvoření tříd pro nové typy záhlaví pomocí IBM MQ classes for Java
Můžete vytvořit třídy Java pro typy záhlaví, které nejsou dodávány s produktem IBM MQ classes for Java.

Chcete-li přidat třídu Java představující nový typ záhlaví, který můžete použít stejným způsobem jako libovolnou třídu záhlaví dodávanou s produktem IBM MQ classes for Java, vytvoříte třídu, která implementuje rozhraní MQHeader. Nejjednodušším přístupem je rozšíření třídy com.ibm.mq.headers.impl.Header . Tento příklad vytvoří plně funkční třídu představující strukturu záhlaví MQTM. Nemusíte přidávat jednotlivé metody getter a setter pro každé pole, ale je to užitečné pro uživatele třídy záhlaví. Generické metody getValue a setValue , které berou řetězec pro název pole, budou fungovat pro všechna pole definovaná v typu záhlaví. Zděděné metody čtení, zápisu a velikosti umožní, aby instance nového typu záhlaví byly čteny a zapisovány, a vypočítají velikost záhlaví správně na základě jeho definice pole. Definice typu je vytvořena pouze jednou, avšak mnoho instancí této třídy záhlaví je vytvořeno. Chcete-li zpřístupnit novou definici záhlaví pro dekódování pomocí tříd MQHeaderIterator nebo MQHeaderList , zaregistrujte 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.
}
```

Zpracování zpráv PCF pomocí 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 komfortní metody pro přidávání a načítání parametrů PCF.

Struktury parametrů PCF jsou reprezentovány třídami MQCFH, MQCFIN, MQCFIN64, MQCFST, MQCFBS, MQCFIL, MQCFIL64 MQCFSL a MQCFGR. Tato sdílena základní provozní rozhraní:

- Metody pro čtení a zápis obsahu zprávy: read (), write () a size ()
- Metody pro manipulaci s parametry: getValue (), setValue (), getParameter () a další
- Metoda enumerator.nextParameter (), která analyzuje obsah PCF v MQMessage

Parametr filtru PCF se používá v příkazech inquire k poskytnutí funkce filtru. Je zapouzdřen v následujících třídách:

- MQCFIF-celočíselný filtr
- MQCFSF-řetězcový filtr
- Filtr bajtů MQCFBF

Ke správě připojení ke správci front, frontě příkazového serveru a přidružené frontě odpovědí jsou poskytnuty dvě třídy agenta, PCFAgent a PCFMessageAgent . PCFMessageAgent rozšiřuje agenta PCFAgent a obvykle by měl být používán přednostně. Třída PCFMessageAgent převede přijaté zprávy MQMessage a předá je zpět volajícímu jako pole PCFMessage. PCFAgent vrací pole zpráv MQMessage, které musíte před použitím analyzovat.

Zpracování vlastností zprávy v souboru IBM MQ classes for Java

Volání funkcí ke zpracování popisovačů zpráv nemají v produktu IBM MQ classes for Javaekvivalent. Chcete-li nastavit, vrátit nebo odstranit vlastnosti popisovače zprávy, použijte metody třídy MQMessage.

Obecné informace o vlastnostech zprávy viz “Názvy vlastností” na stránce 27.

V produktu IBM MQ classes for Java je přístup ke zprávám prostřednictvím třídy MQMessage. Proto nejsou v prostředí Java poskytnuty manipulátory zpráv a neexistuje žádný ekvivalent pro volání funkce IBM MQ MQCRTMH, MQDLTMH, MQMHBUF a MQBUFMH.

Chcete-li nastavit vlastnosti manipulátoru zpráv v procedurálním rozhraní, použijte volání MQSETMP. V produktu IBM MQ classes for Javapoužijte příslušnou metodu třídy MQMessage:

- setBooleanVlastnost
- Vlastnost setByte
- Vlastnost setBytes
- Vlastnost setShort
- Vlastnost setInt
- setInt2Property
- setInt4Property
- setInt8Property
- Vlastnost setLong
- Vlastnost setFloat
- Vlastnost setDouble
- Vlastnost setString
- Vlastnost setObject

Někdy se na ně souhrnně odkazuje jako na metody *set*property* .

Chcete-li vrátit hodnotu vlastností manipulátoru zprávy v procedurálním rozhraní, použijte volání MQINQMP. V produktu IBM MQ classes for Javapoužijte příslušnou metodu třídy MQMessage:

- getBooleanVlastnost
- Vlastnost getByte
- Vlastnost getBytes
- Vlastnost getShort
- Vlastnost getInt
- getInt2Property
- getInt4Property
- getInt8Property
- Vlastnost getLong
- Vlastnost getFloat
- Vlastnost getDouble

- Vlastnost getString
- Vlastnost getObject

Na tyto metody se někdy souhrnně odkazuje jako na metody *get*property* .

Chcete-li odstranit hodnotu vlastností manipulátoru zprávy v procedurálním rozhraní, použijte volání MQDLTMP. V produktu IBM MQ classes for Javapoužijte metodu deleteProperty třídy MQMessage.

Zpracování chyb v souboru IBM MQ classes for Java

Ošetřit chyby vzniklé v důsledku IBM MQ classes for Java použití bloků Java `try` a `catch` .

Metody v rozhraní Java nevracejí kód dokončení a kód příčiny. Namísto toho vygenerují výjimku vždy, když kód dokončení a kód příčiny, který je výsledkem volání IBM MQ , nejsou obě nula. To zjednodušuje logiku programu, abyste nemuseli kontrolovat návratové kódy po každém volání IBM MQ. Můžete se rozhodnout, na kterých místech ve vašem programu se chcete vypořádat s možností selhání. V těchto bodech můžete kód obklopit bloky `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() );
}
```

IBM MQ kódy příčiny volání nahlášené zpět v Java výjimkách pro z/OS jsou zdokumentovány v Kódu příčiny a dokončení rozhraní API.

Do protokolu se také zapisují výjimky, které se objeví v době, kdy je spuštěna aplikace IBM MQ classes for Java . Aplikace však může volat metodu MQException.logExclude(), aby zabránila protokolování výjimek přidružených ke specifickému kódu příčiny. To můžete provést v situacích, kdy očekáváte mnoho výjimek přidružených ke specifickému kódu příčiny a nechcete, aby byl protokol vyplněn těmito výjimkami. Pokud se například vaše aplikace pokusí získat zprávu z fronty pokaždé, když iteruje kolem smyčky, a u většiny těchto pokusů očekáváte, že ve frontě nebude žádná vhodná zpráva, můžete zabránit protokolování výjimek přidružených ke kódu příčiny MQRC_NO_MSG_AVAILABLE. Pokud aplikace dříve zabránila protokolování výjimek přidružených ke specifickému kódu příčiny, může tyto výjimky znovu protokolovat voláním metody MQException.logInclude().

Někdy kód příčiny nepředává všechny podrobnosti související s chybou. Pro každou vyvolanou výjimku by měla aplikace zkontrolovat připojenou výjimku. Samotná propojená výjimka může mít jinou propojenou výjimku, a proto propojené výjimky tvoří řetězec vedoucí zpět k původnímu základnímu problému. Připojená výjimka je implementována pomocí mechanizmu zřetězených výjimek třídy java.lang.Throwable a aplikace obdrží propojenou výjimku voláním metody Throwable.getCause(). Z výjimky, která je instancí výjimky MQException, funkce MQException.getCause() načte základní instanci com.ibm.mq.jmqi.JmqiExceptiona funkce getCause z této výjimky načte základní java.lang.Exception , která způsobila chybu.

Získání a nastavení hodnot atributů v souboru IBM MQ classes for Java

Metody getXXX() a setXXX() jsou poskytovány pro mnoho společných atributů. K ostatním lze přistupovat pomocí generických metod inquire () a set ().

Pro mnoho společných atributů obsahují třídy MQManagedObject, MQDestination, MQQueue, MQTopic, MQProcess a MQQueueManager metody getXXX() a setXXX(). Tyto metody vám umožňují získat a nastavit jejich hodnoty atributů. Všimněte si, že pro MQDestination, MQQueue a MQTopic fungují metody pouze v případě, že při otevření objektu zadáte příslušné příznaky inquire a set.

U méně běžných atributů všechny třídy MQQueueManager, MQDestination, MQQueue, MQTopic a MQProcess dědí z třídy s názvem MQManagedObject. Tato třída definuje rozhraní inquire () a set ().

Když vytvoříte nový objekt správce front pomocí operátoru *new* , automaticky se otevře pro zjišťování. Když použijete metodu accessProcess() pro přístup k objektu procesu, tento objekt se automaticky otevře pro zjišťování. Použijete-li metodu accessQueue() pro přístup k objektu fronty, nebude tento objekt automaticky otevřen pro operace dotazu nebo nastavení. Důvodem je, že automatické přidání těchto voleb může způsobit problémy s některými typy vzdálených front. Chcete-li ve frontě použít metody inquire, set, getXXXa setXXX , musíte v parametru openOptions metody accessQueue() zadat příslušné příznaky inquire a set. Totéž platí pro cílové objekty a objekty témat.

Metody inquire a set používají 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 známa. 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));
```

Programy s podporou podprocesů v adresáři Java

Běhové prostředí Java je ze své podstaty vícevláknové. Produkt IBM MQ classes for Java umožňuje sdílení objektu správce front více podprocesy, ale zajišťuje synchronizaci veškerého přístupu k cílovému správci front.

Vícevláknové programy se v produktu Javatěžko vyvarují. 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 Java je ze své podstaty vícevláknové. Proto dojde k inicializaci aplikace v jednom podprocesu a kód, který se provádí jako odpověď na tlačítko, se spustí v samostatném podprocesu (podproces uživatelského rozhraní).

V případě systému IBM MQ MQI clientzaloženého na jazyku C by to způsobilo problém, protože existuje omezení sdílení manipulátorů více podprocesy. Produkt IBM MQ classes for Java uvolní toto omezení a umožní sdílení objektu správce front (a jeho přidružených objektů front, témat a procesů) více podprocesy.

Implementace IBM MQ classes for Java zajišťuje, že pro konkrétní připojení (instance objektuMQQueueManager) je synchronizován veškerý přístup k cílovému správci front IBM MQ . Podproces, který chce zadat volání do správce front, je blokován, dokud nebudou dokončena všechna ostatní probíhající volání pro toto připojení. Pokud vyžadujete souběžný přístup ke stejnému správci front z více podprocesů v rámci programu, vytvořte nový objekt MQQueueManager pro každý podproces, který vyžaduje souběžný přístup. (Jedná se o ekvivalent k zadání samostatného volání MQCONN pro každý podproces.)

Poznámka: Instance třídy `com.ibm.mq.MQGetMessageOptions` nesmí být sdíleny mezi podprocesy, které požadují zprávy souběžně. Instance této třídy jsou aktualizovány daty během příslušného požadavku MQGET, což může vést k neočekávaným důsledkům v případě, že více podprocesů souběžně pracuje na stejné instanci objektu.

Použití uživatelských procedur kanálu v adresáři IBM MQ classes for Java

Přehled toho, jak používat uživatelské procedury kanálu v aplikaci pomocí konzoly IBM MQ classes for Java.

Následující témata popisují, jak napsat uživatelskou proceduru kanálu v produktu Java, jak ji přiřadit a jak jí předat data. Dále popisují, jak používat uživatelské procedury kanálu napsané v jazyce C a jak používat posloupnost uživatelských procedur kanálu.

Vaše aplikace musí mít správné oprávnění zabezpečení pro načtení třídy uživatelské procedury kanálu.

Vytvoření uživatelské procedury kanálu v adresáři IBM MQ classes for Java

Můžete poskytnout vlastní uživatelské procedury kanálu definováním třídy Java , která implementuje příslušné rozhraní.

Chcete-li implementovat uživatelskou proceduru, definujte novou třídu Java , která implementuje příslušné rozhraní. V balíku com.ibm.mq.exits jsou definována tři rozhraní uživatelské procedury:

- WMQSendExit
- WMQReceiveExit
- WMQSecurityExit

Poznámka: Uživatelské procedury kanálu jsou podporovány pouze pro připojení klienta; nejsou podporovány pro připojení vazeb. Nelze použít Java uživatelskou proceduru kanálu mimo IBM MQ classes for Java, například pokud používáte klientskou aplikaci napsanou v jazyce C.

Jakékoli šifrování TLS definované pro připojení je provedeno *po* vyvolání uživatelských procedur pro odeslání a zabezpečení. Podobně se dešifrování provádí *před* vyvoláním uživatelských procedur pro příjem a 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 představují struktury MQCXP a MQCD definované v procedurálním rozhraní.

Každá třída ukončení, kterou napíšete, musí mít konstruktor. Může se jednat o výchozí konstruktor nebo o konstruktor, který používá řetězcový argument. Pokud vezme řetězec, uživatelská data budou předána do třídy ukončení při jejím vytvoření. Pokud třída ukončení obsahuje jak výchozí konstruktor, tak i konstruktor s jedním argumentem, má konstruktor s jedním argumentem prioritu.

Pro uživatelské procedury odeslání a zabezpečení musí kód ukončení vrátit data, která chcete odeslat na server. V případě uživatelské procedury pro příjem musí kód uživatelské procedury vrátit upravená data, která má produkt IBM MQ interpretovat.

Nejjednodušší možné výstupní těleso je:

```
{ return agentBuffer; }
```

Nezavírejte správce front z uživatelské procedury kanálu.

Použití existujících tříd ukončení kanálu

Ve verzích produktu IBM MQ starších než 7.0byste tyto uživatelské procedury implementovali pomocí rozhraní MQSendExit, MQReceiveExita MQSecurityExit, jako v následujícím příkladu. Tato metoda zůstává platná, ale pro lepší funkčnost a výkon je preferována nová metoda.

```
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 uživatelské procedury kanálu v adresáři IBM MQ classes for Java
Uživatelskou proceduru kanálu můžete přiřadit pomocí IBM MQ classes for Java.

Neexistuje žádný přímý ekvivalent kanálu IBM MQ v souboru IBM MQ classes for Java. Uživatelské procedury kanálu jsou přiřazeny ke správci MQQueueManager. Například po definování třídy, která implementuje rozhraní WMQSecurityExit , může aplikace použít uživatelskou proceduru zabezpečení jedním ze čtyř způsobů:

- Přiřazením instance třídy k poli MQEnvironment.channelSecurityExit před vytvořením objektu MQQueueManager .
- Nastavením pole MQEnvironment.channelSecurityExit na řetězec představující třídu uživatelské procedury zabezpečení před vytvořením objektu MQQueueManager .
- Vytvořením dvojice klíč/hodnota v hašovací tabulce vlastností předané správci MQQueueManager s klíčem CMQC.SECURITY_EXIT_PROPERTY
- Použití tabulky CCDT (Client Channel Definition Table)

Jakákoli uživatelská procedura přiřazená nastavením pole MQEnvironment.channelSecurityExit na řetězec, vytvořením dvojice klíč/hodnota v hašovací tabulce vlastností nebo pomocí tabulky CCDT musí být zapsána s výchozím konstruktorem. Uživatelská procedura přiřazená jako instance třídy nepotřebuje v závislosti na aplikaci výchozí konstruktor.

Aplikace může použít odeslání nebo ukončení příjmu podobným způsobem. Například následující fragment kódu ukazuje, jak používat uživatelské procedury zabezpečení, odeslání a přijetí, které jsou implementovány 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, pořadí priorit je následující:

1. Je-li URL tabulky CCDT předána správci MQQueueManager, obsah tabulky CCDT určuje uživatelské procedury kanálu, které mají být použity, a všechny definice uživatelských procedur v prostředí MQEnvironment nebo hašovací tabulky vlastností jsou ignorovány.
2. Pokud není předána žádná URL tabulky CCDT, sloučí se definice ukončení z prostředí MQEnvironment a hašovací tabulky.
   - Je-li v prostředí MQEnvironment i v hašovací tabulce definován stejný typ uživatelské procedury, použije se definice v hašovací tabulce.
   - Pokud jsou zadány ekvivalentní staré a nové typy uživatelské procedury (například pole sendExit , které lze použít pouze pro typ uživatelské procedury použitý ve verzích starších než IBM WebSphere MQ 7.0, a pole uživatelské procedury channelSend, které lze použít pro libovolnou uživatelskou proceduru odeslání), použije se nová uživatelská procedura (channelSendExit) spíše než stará uživatelská procedura.

Pokud jste deklarovali uživatelskou proceduru kanálu jako řetězec, musíte povolit, aby IBM MQ vyhledal program uživatelské procedury kanálu. Můžete tak učinit různými způsoby v závislosti na prostředí, ve kterém je aplikace spuštěna, a na tom, jak jsou programy uživatelské procedury kanálu zabaleny.

- Pro aplikaci, která je spuštěna na aplikačním serveru, musíte uložit soubory v adresáři uvedeném v souboru Tabulka 58 na stránce 380 nebo zabaleném v souborech JAR, na které odkazuje **exitClasspath**.
- Pro aplikaci, která není spuštěna na aplikačním serveru, platí následující pravidla:
  - Pokud jsou vaše třídy uživatelské procedury kanálu zabaleny do samostatných souborů JAR, musí být tyto soubory JAR zahrnuty do souboru **exitClasspath**.
  - Nejsou-li třídy uživatelských procedur kanálu zabaleny v souborech JAR, mohou být soubory tříd uloženy v adresáři zobrazeném v adresáři Tabulka 58 na stránce 380 nebo v libovolném adresáři v systémové cestě ke třídám prostředí JVM nebo v adresáři **exitClasspath**.

Vlastnost **exitClasspath** lze zadat čtyřmi způsoby. V pořadí podle priority jsou tyto způsoby následující:

1. Systémová vlastnost com.ibm.mq.exitClasspath (definovaná na příkazovém řádku pomocí volby `-D`).
2. Sekce exitPath souboru `mqclient.ini`
3. Položka hašovací tabulky s klíčem CMQC.EXIT_CLASSPATH_PROPERTY
4. Proměnná prostředí MQEnvironment **exitClasspath**

Oddělte více cest pomocí znaku java.io.File.pathSeparator .

Tabulka 58. Adresář pro programy uživatelských procedur kanálu

Platforma	**Adresář**
AIX AIX Linux Linux	`/var/mqm/exits` (32bitové uživatelské programy kanálu) `/var/mqm/exits64` (64bitové uživatelské programy kanálu)
Windows	*instalační_datový_adresář*\exits

Poznámka: *install_data_dir* je adresář, který jste vybrali pro datové soubory IBM MQ během instalace. Výchozí adresář je `C:\ProgramData\IBM\MQ`.

Předávání dat do uživatelských procedur kanálu v adresáři IBM MQ classes for Java
Data můžete předat do uživatelských procedur kanálu a vrátit data z uživatelských procedur kanálu do aplikace.

Parametr agentBuffer

Pro uživatelskou proceduru odeslání obsahuje parametr *agentBuffer* data, která se mají odeslat. V případě uživatelské procedury pro příjem nebo uživatelské procedury zabezpečení obsahuje parametr *agentBuffer* data, která byla právě přijata. Parametr délky nepotřebujete, protože výraz agentBuffer.limit () označuje délku pole.

Pro uživatelské procedury odeslání a zabezpečení musí kód ukončení vrátit data, která chcete odeslat na server. V případě uživatelské procedury pro příjem musí kód uživatelské procedury vrátit upravená data, která má produkt IBM MQ interpretovat.

Nejjednodušší možné výstupní těleso je:

```
{ return agentBuffer; }
```

Uživatelské procedury kanálu jsou volány s vyrovnávací pamětí, která má záložní pole. Pro dosažení nejlepšího výkonu by měla uživatelská procedura vrátit vyrovnávací paměť s podpůrným polem.

Data uživatele

Pokud se aplikace připojí ke správci front nastavením uživatelských procedur channelSecurityExit, channelSendExit nebo channelReceiveExit, lze 32 bajtů uživatelských dat předat příslušné třídě uživatelské procedury kanálu při volání pomocí polí channelSecurityExitUserData, channelSendExitUserData nebo channelReceiveExitUserData. Tato uživatelská data jsou k dispozici pro třídu uživatelské procedury kanálu, ale jsou aktualizována při každém volání uživatelské procedury. Veškeré 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 jsou udržována mezi vyvoláními uživatelské procedury.

Pokud aplikace nastaví securityExit, sendExitnebo receiveExit, nelze těmto třídám uživatelské procedury kanálu předat žádná data.

Pokud aplikace pro připojení ke správci front používá tabulku CCDT (Client Channel Definition Table), jsou při volání předána třídám uživatelských procedur kanálu veškerá uživatelská data určená v definici kanálu připojení klienta. Další informace o použití tabulky definic kanálů klienta viz “Použití tabulky definic kanálů klienta s volbou IBM MQ classes for Java” na stránce 364.

Použití uživatelských procedur kanálu, které nejsou zapsány v souboru Java , s IBM MQ classes for Java
Jak používat programy uživatelské procedury kanálu napsané v jazyce C z aplikace Java .

V produktu IBM MQmůžete zadat název programu uživatelské procedury kanálu zapsaného v jazyce C jako řetězec předaný do polí uživatelské procedury channelSecurity, uživatelské procedury channelSendnebo channelReceivev objektu MQEnvironment nebo v hašovací tabulce vlastností. Nelze však použít uživatelskou proceduru kanálu napsanou v adresáři Java v aplikaci napsané v jiném jazyce.

Uveďte název uživatelského programu ve formátu `library(function)` a ujistěte se, že umístění uživatelského programu je uvedeno, jak je popsáno v části Cesta k ukončení.

Informace o zápisu uživatelské procedury kanálu v jazyce C naleznete v části “Programy uživatelské procedury kanálu pro kanály systému zpráv” na stránce 928.

Použití posloupnosti uživatelských procedur pro odeslání nebo příjem kanálu v adresáři IBM MQ classes for Java
Aplikace IBM MQ classes for Java může používat posloupnost uživatelských procedur pro odeslání nebo příjem kanálu, které jsou spouštěny v posloupnosti.

Chcete-li použít posloupnost uživatelských procedur odeslání, může aplikace vytvořit seznam nebo řetězec obsahující uživatelské procedury odeslání. Je-li použit seznam, každý prvek seznamu může být některý z následujících:

- Instance třídy definované uživatelem, která implementuje rozhraní WMQSendExit .
- Instance třídy definované uživatelem, která implementuje rozhraní MQSendExit (pro uživatelskou proceduru odeslání napsanou v souboru Java).
- Instance výstupní třídy MQExternalSend(pro uživatelskou proceduru odeslání nezapsanou v souboru Java).
- Instance třídy řetězu MQSendExit
- Instance třídy String

Seznam nemůže obsahovat jiný seznam.

Aplikace může používat posloupnost uživatelských procedur příjmu podobným způsobem.

Je-li použit řetězec, musí se skládat z jedné nebo více definic ukončení oddělených čárkami, z nichž každá může být názvem třídy Java nebo programem v jazyce C ve formátu `library(function)`.

Aplikace poté přiřadí objekt List nebo String k poli MQEnvironment.channelSendExit před vytvořením objektu MQQueueManager .

Kontext informací předávaných východům je výhradně v rámci domény těchto východů. Pokud jsou například uživatelská procedura Java a uživatelská procedura C zřetězeny, nemá přítomnost uživatelské procedury Java žádný vliv na uživatelskou proceduru C.

Použití tříd výstupního řetězce

Ve verzích starších než IBM WebSphere MQ 7.0byly poskytnuty dvě třídy, které povolovaly posloupnosti uživatelských procedur:

- Řetězec MQSendExit, který implementuje rozhraní MQSendExit .
- Řetězec MQReceiveExit, který implementuje rozhraní MQReceiveExit .

Použití těchto tříd zůstává platné, ale je preferována nová metoda. Použití rozhraní IBM MQ Třídy pro Java znamená, že vaše aplikace má stále závislost na `com.ibm.mq.jar` Pokud se používá nová sada rozhraní v balíku `com.ibm.mq.exits` , není závislost na `com.ibm.mq.jar`.

Chcete-li použít posloupnost uživatelských procedur odeslání, aplikace vytvořila seznam objektů, kde každý objekt byl jeden z následujících:

- Instance třídy definované uživatelem, která implementuje rozhraní MQSendExit (pro uživatelskou proceduru odeslání napsanou v souboru Java).
- Instance výstupní třídy MQExternalSend(pro uživatelskou proceduru odeslání nezapsanou v souboru Java).
- Instance třídy řetězu MQSendExit

Aplikace vytvořila objekt řetězce MQSendExitpředáním tohoto seznamu objektů jako parametru konstruktoru. Aplikace by pak před vytvořením objektu MQQueueManager přiřadila objekt řetězce MQSendExitk poli MQEnvironment.sendExit .

Komprese kanálů v souboru IBM MQ classes for Java

Komprimace dat, která procházejí kanálem, může zlepšit výkon kanálu a snížit provoz na síti. IBM MQ classes for Java použijte funkci komprese vestavěnou do IBM MQ.

Pomocí funkce dodávané s produktem IBM MQmůžete komprimovat data, která proudí v kanálech zpráv a kanálech MQI, a v obou typech kanálů můžete komprimovat data záhlaví a data zpráv nezávisle na sobě.

Při výchozím nastavení nejsou na kanálu komprimována žádná data. Úplný popis komprese kanálu včetně toho, jak je implementována v produktu IBM MQ, viz Komprese dat (COMPMSG) a Komprese záhlaví (COMPHDR).

Aplikace IBM MQ classes for Java určuje techniky, které lze použít pro komprimaci dat záhlaví nebo zprávy v připojení klienta vytvořením objektu java.util.Collection . Každá technika komprese je celočíselný objekt v kolekci a pořadí, ve kterém aplikace přidává techniky komprese do kolekce, je pořadí, ve kterém jsou techniky komprese vyjednávány se správcem front při spuštění připojení klienta. Aplikace pak může přiřadit kolekci k poli hdrCompList pro data záhlaví nebo k poli msgCompList pro data zprávy ve třídě MQEnvironment. Jakmile je aplikace připravena, může spustit připojení klienta vytvořením objektu MQQueueManager .

Následující fragmenty kódu ilustrují 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 zpráv:

```
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 vyjednány v pořadí RLE, pak ZLIBHIGH, když se spustí připojení klienta. Vybranou techniku komprese nelze změnit během doby životnosti objektu MQQueueManager .

Techniky komprese pro data záhlaví a zprávy, která jsou podporována klientem i správcem front v připojení klienta, jsou předávány uživatelské proceduře kanálu jako kolekce v polích Seznam hdrCompa Seznam msgCompobjektu MQChannelDefinition . Skutečné techniky, které se aktuálně používají pro kompresi dat záhlaví a zprávy v připojení klienta, jsou předány uživatelské proceduře kanálu v polích CurHdrKomprese a CurMsgKomprese objektu MQChannelExit .

Je-li pro připojení klienta použita komprese, jsou data komprimována před zpracováním a extrakcí uživatelských procedur odeslání kanálu po zpracování uživatelských procedur příjmu kanálu. Data předaná k odeslání a přijetí uživatelských procedur jsou proto v komprimovaném stavu.

Další informace o určování technik komprese a o dostupných technikách komprese naleznete v tématu Třída com.ibm.mq.MQEnvironment a Rozhraní com.ibm.mq.MQC.

Sdílení připojení TCP/IP v produktu IBM MQ classes for Java

Pro sdílení jednoho připojení TCP/IP lze vytvořit více instancí kanálu MQI.

V produktu IBM MQ classes for Javamůžete pomocí proměnné MQEnvironment.sharingConversations řídit počet konverzací, které mohou sdílet jedno připojení TCP/IP.

Atribut SHARECNV představuje nejlepší přístup ke sdílení připojení. Proto není při použití hodnoty SHARECNV větší než 0 s hodnotou IBM MQ classes for Java zaručeno, že nový požadavek na připojení bude vždy sdílet již zavedené připojení.

Sdružování připojení v adresáři IBM MQ classes for Java

Produkt IBM MQ classes for Java umožňuje uložit do fondu náhradní připojení pro opětovné použití.

Produkt IBM MQ classes for Java poskytuje další podporu pro aplikace, které se zabývají více připojeními ke správcům front produktu IBM MQ . Pokud již připojení není vyžadováno, lze jej namísto jeho zničení

sdružit do fondu a později znovu použít. To může poskytnout podstatné zvýšení výkonu pro aplikace a middleware, které se sériově připojují k libovolným správcům front.

Produkt IBM MQ poskytuje výchozí fond připojení. Aplikace mohou aktivovat nebo deaktivovat tento fond připojení registrací a zrušením registrace tokenů prostřednictvím třídy MQEnvironment. Pokud je fond aktivní, když produkt IBM MQ classes for Java vytvoří objekt MQQueueManager , prohledá tento výchozí fond a znovu použije vhodná připojení. Dojde-li k volání MQQueueManager.disconnect (), vrátí se základní připojení do fondu.

Aplikace mohou také vytvořit fond připojení MQSimpleConnectionManager pro konkrétní použití. Poté může aplikace buď určit fond během konstrukce objektu MQQueueManager , nebo tento fond předat prostředí MQEnvironment pro použití jako výchozí fond připojení.

Chcete-li zabránit připojení v použití příliš velkého množství prostředků, můžete omezit celkový počet připojení, která může objekt správce MQSimpleConnectionManager zpracovat, a omezit velikost fondu připojení. Nastavení limitů je užitečné, pokud existují konfliktní požadavky na připojení v rámci prostředí JVM.

Standardně metoda getMaxConnections () vrací hodnotu nula, což znamená, že neexistuje žádné omezení počtu připojení, která může objekt MQSimpleConnectionManager zpracovat. Limit můžete nastavit pomocí metody setMaxConnections (). Pokud nastavíte limit a limit je dosažen, může požadavek na další připojení způsobit vyvolání výjimky MQException s kódem příčiny MQRC_MAX_CONNS_LIMIT_REACHED.

Řízení výchozího fondu připojení v adresáři IBM MQ classes for Java
Tento příklad ukazuje, jak použít výchozí fond připojení.

Zvažte následující ukázku aplikace MQApp1:

```
import com.ibm.mq.*;
public class MQApp1
{
      public static void main(String[] args) throws MQException
      {
         for (int i=0; i<args.length; i++) {
            MQQueueManager qmgr=new MQQueueManager(args[i]);
            :
            : (do something with qmgr)
            :
            qmgr.disconnect();
         }
      }
}
```

MQApp1 vezme seznam lokálních správců front z příkazového řádku, připojí se ke každému z nich a provede určitou operaci. Pokud však příkazový řádek vypisuje stejného správce front vícekrát, je efektivnější připojit se pouze jednou a opakovaně používat toto připojení.

Produkt IBM MQ classes for Java poskytuje výchozí fond připojení, který můžete použít k tomuto použití. Chcete-li fond povolit, použijte jednu z metod MQEnvironment.addConnectionPoolToken(). Chcete-li fond zakázat, použijte funkci MQEnvironment.removeConnectionPoolToken().

Následující ukázková aplikace MQApp2je funkčně identická s aplikací MQApp1, ale připojuje se ke každému správci 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 tučný řádek aktivuje výchozí fond připojení registrací objektu MQPoolToken v prostředí MQEnvironment.

Konstruktor MQQueueManager nyní vyhledá v tomto fondu příslušné připojení a vytvoří připojení ke správci front pouze v případě, že nemůže najít existující připojení. Volání qmgr.disconnect() vrátí připojení do fondu pro pozdější opětovné 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 jinak by aplikace byla ukončena s počtem aktivních připojení správce front ve fondu. Tato situace může způsobit chyby, které se objeví v protokolech správce front.

Pokud aplikace používá tabulku CCDT (Client Channel Definition Table) pro připojení ke správci front, konstruktor MQQueueManager nejprve vyhledá v tabulce vhodnou definici kanálu připojení klienta. Je-li nalezen, konstruktor vyhledá výchozí fond připojení pro připojení, které lze použít pro kanál. Pokud konstruktor nemůže najít vhodné připojení ve fondu, vyhledá v tabulce definic kanálů klienta další vhodnou definici kanálu připojení klienta a pokračuje podle výše uvedeného popisu. Pokud konstruktor dokončí hledání tabulky definic kanálů klienta a nenajde ve fondu žádné vhodné připojení, konstruktor spustí druhé hledání tabulky. Během tohoto hledání se konstruktor pokusí vytvořit nové připojení pro každou vhodnou definici kanálu připojení klienta a použije první připojení, které spravuje.

Výchozí fond připojení ukládá maximálně deset nepoužívaných připojení a ponechává nepoužívaná připojení aktivní po dobu maximálně pěti minut. Aplikace to může změnit (podrobnosti viz “Dodání jiného fondu připojení v adresáři IBM MQ classes for Java” na stránce 386).

Namísto použití prostředí MQEnvironment k dodání tokenu MQPoolTokenmůže aplikace vytvořit vlastní:

```
    MQPoolToken token=new MQPoolToken();
    MQEnvironment.addConnectionPoolToken(token);
```

Některé aplikace nebo dodavatelé middlewaru poskytují podtřídy MQPoolToken , aby předali informace do vlastního fondu připojení. Lze je sestavit a předat do funkce addConnectionPoolToken() tímto 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 MQPoolTokens ze statické sady registrovaných objektů MQPoolToken .

Prostředí MQEnvironment obsahuje statickou sadu registrovaných objektů MQPoolToken . Chcete-li přidat nebo odebrat MQPoolTokens z této sady, použijte následující metody:

- MQEnvironment.addConnectionPoolToken()
- MQEnvironment.removeConnectionPoolToken()

Aplikace se může skládat z mnoha komponent, které existují nezávisle a provádějí práci pomocí správce front. V takové aplikaci by každá komponenta měla přidat token MQPoolToken do sady MQEnvironment po dobu své životnosti.

Například vzorová aplikace MQApp3 vytvoří deset podprocesů a spustí každý z nich. Každý podproces registruje svůj vlastní token MQPoolToken, čeká určitou dobu a poté se připojí ke správci front. Poté, co se podproces odpojí, odebere svůj vlastní MQPoolToken.

Výchozí fond připojení zůstává aktivní, pokud je v sadě MQPoolTokensalespoň 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 adresáři IBM MQ classes for Java
Tento příklad ukazuje, jak použít třídu **com.ibm.mq.MQSimpleConnectionManager** k dodání jiného fondu připojení.

Tato třída poskytuje základní prostředky pro sdružování připojení a aplikace mohou tuto třídu použít k přizpůsobení chování fondu.

Po vytvoření instance lze v konstruktoru MQQueueManager zadat správce MQSimpleConnection. Správce MQSimpleConnectionpoté spravuje připojení, které je základem sestavené komponenty MQQueueManager. Pokud MQSimpleConnectionManager obsahuje vhodné připojení ve fondu, bude toto připojení znovu použito a vráceno do správce MQSimpleConnectionManager po volání 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 myConnMan po volání qmgr.disconnect(). Připojení je poté znovu použito během druhého volání konstruktoru MQQueueManager .

Druhý řádek povoluje správce MQSimpleConnection. Poslední řádek zakáže správce MQSimpleConnectionManager a zničí veškerá připojení ve fondu. Správce MQSimpleConnectionje standardně v MODE_AUTO, což je popsáno dále v této části.

Správce MQSimpleConnectionpřiděluje připojení na základě nejnovějšího použití a likviduje připojení na základě nejnovějšího použití. Při výchozím nastavení je připojení zničeno, pokud nebylo použito po dobu pěti minut nebo pokud je ve fondu více než deset nepoužívaných připojení. Tyto hodnoty můžete změnit voláním funkce MQSimpleConnectionManager.setTimeout().

Můžete také nastavit správce MQSimpleConnectionpro použití jako výchozí fond připojení, který bude použit v případě, že v konstruktoru MQQueueManager není uveden žádný správce Connection Manager.

Následující aplikace to demonstruje:

```
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 správce MQSimpleConnection. Konfigurace provádí následující akce:

- Ukončí připojení, která nejsou používána po dobu jedné hodiny.
- Omezuje počet připojení spravovaných produktem myConnMan na 75
- Omezuje počet nevyužitých připojení ve fondu na 50
- Nastaví MODE_AUTO, což je výchozí hodnota. To znamená, že fond je aktivní pouze v případě, že se jedná o výchozího správce připojení, a v sadě MQPoolTokens , kterou drží prostředí MQEnvironment, je alespoň jeden token.

Nový správce MQSimpleConnectionManager je poté nastaven jako výchozí správce připojení.

V posledním řádku aplikace volá MQApp3.main(). To spustí několik podprocesů, kde každý podproces používá IBM MQ nezávisle. Tyto podprocesy používají při vytváření připojení příkaz myConnMan.

JTA/JDBC koordinace pomocí 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 kompatibilní s rozhraním JDBC typu 2 nebo JDBC typu 4.

Tato podpora není k dispozici na všech platformách. Chcete-li zkontrolovat, které platformy podporují koordinaci JDBC , viz 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 ostatních platforem.

Konfigurace koordinace JTA/JDBC na Windows

Knihovna XA je dodávána jako knihovna DLL s názvem ve formátu `jdbcxxx.dll`.

Dodaný produkt `jdbcora12.dll` poskytuje kompatibilitu s produktem Oracle 12Cpro instalaci serveru IBM MQ for 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` vaší instalace produktu IBM MQ classes for Java . Musíte deklarovat knihovnu XA, která je také popsána jako soubor načtení přepínače, do správce front. Použijte IBM MQ Explorer. Zadejte podrobnosti o souboru načtení přepínače na panelu vlastností správce front pod správcem prostředků XA. Musíte zadat pouze název knihovny. Příklad:

Pro databázi Db2 nastavte pole SwitchFile na: dbcdb2

Pro databázi Oracle nastavte pole SwitchFile na: jdbcora

Notes:

1. Oracle 12C je podporován IBM MQ classes for Java, pouze na IBM MQ pro Windows.
2. Podporovaná verze produktu Oracle 12C je 12.1.0.1.0 Enterprise Edition a budoucí opravné sady.
3. Oracle 64bitové databáze na 64bitových Windows vyžadují 32bitového klienta Oracle .
4. Pomocí konzoly IBM MQ classes for Javamůže produkt IBM MQ vystupovat jako koordinátor transakcí. Není však možné se účastnit transakce ve stylu JTA.

Konfigurace koordinace JTA/JDBC na jiných platformách než Windows
Jsou dodány soubory objektů. Propojte příslušný soubor pomocí dodaného souboru Makefile a deklarujte jej pro správce front pomocí konfiguračního souboru.

Pro každý systém správy databází poskytuje produkt IBM MQ dva soubory objektů. Chcete-li vytvořit 32bitovou knihovnu přepínačů, musíte propojit jeden soubor objektů a propojit druhý soubor objektů, abyste vytvořili 64bitovou knihovnu přepínačů. Pro parametr Db2je název každého souboru objektu jdbcdb2.oa pro parametr Oracle je název každého souboru objektu jdbcora.o.

Každý soubor objektu musíte propojit pomocí příslušného souboru Makefile dodaného s produktem IBM MQ. Knihovna přepínače vyžaduje jiné knihovny, které mohou být uloženy v různých umístěních na různých systémech. Knihovna přepínače však nemůže k vyhledání těchto knihoven použít proměnnou prostředí cesty ke knihovně, protože knihovna přepínače je načtena správcem front, který je spuštěn v prostředí setuid . Dodaný soubor Makefile proto zajišťuje, že knihovna přepínače obsahuje úplné názvy cest k těmto knihovnám.

Chcete-li vytvořit knihovnu přepínačů, zadejte příkaz **make** v následujícím formátu. Chcete-li vytvořit 32bitovou knihovnu přepínačů, zadejte příkaz do adresáře /java/lib/jdbc instalace produktu IBM MQ . Chcete-li vytvořit 64bitovou knihovnu přepínačů, zadejte příkaz do adresáře /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.

Poznámka:

- Chcete-li spustit 32bitové aplikace, musíte vytvořit 32bitovou i 64bitovou knihovnu přepínačů pro každý používaný systém správy databází. Chcete-li spustit 64bitové aplikace, musíte vytvořit pouze 64bitovou knihovnu přepínačů. Pro systém Db2je název každé knihovny přepínačů jdbcdb2 a pro systém Oracleje název každé knihovny přepínačů jdbcora. Soubory Makefile zajišťují, že 32bitové a 64bitové knihovny přepínačů jsou uloženy v různých adresářích IBM MQ . 32bitová knihovna přepínačů je uložena v adresáři /java/lib/jdbc a 64bitová knihovna přepínačů je uložena v adresáři /java/lib64/jdbc .
- Protože můžete nainstalovat produkt Oracle kdekoli v systému, soubory Makefile používají proměnnou prostředí **ORACLE_HOME** k vyhledání umístění, kde je nainstalován produkt Oracle .
- Pokud je produkt IBM MQ nainstalován do jiného než výchozího umístění, změňte hodnotu parametru **MQ_INSTALLATION_PATH** v souboru Makefile.

Po vytvoření knihoven přepínačů pro Db2, Oraclenebo obojí je třeba tyto knihovny deklarovat ve správci front. Pokud konfigurační soubor správce front (qm.ini) již obsahuje sekce XAResourceManager pro databáze Db2 nebo Oracle , musíte nahradit položku SwitchFile v každé sekci jednou z následujících položek:

Pro databázi Db2

```
SwitchFile=jdbcdb2
```

Pro databázi Oracle

```
SwitchFile=jdbcora
```

Neuvádějte úplný název cesty buď 32bitové, nebo 64bitové knihovny přepínačů. Uveďte pouze název knihovny.

Pokud konfigurační soubor správce front dosud neobsahuje sekce `XAResourceManager` pro databáze Db2 nebo Oracle nebo pokud chcete přidat další sekce `XAResourceManager` , prohlédněte si část Administrace IBM MQ , kde naleznete informace o tom, jak vytvořit sekci `XAResourceManager` . Avšak každá položka `SwitchFile` v nové sekci `XAResourceManager` musí být přesně tak, jak bylo popsáno dříve pro databázi Db2 nebo Oracle . Musíte také zahrnout položku `ThreadOfControl=PROCESS`.

Po aktualizaci konfiguračního souboru správce front a po nastavení všech příslušných proměnných prostředí databáze můžete správce front restartovat.

Použití koordinace JTA/JDBC
Kódovat volání rozhraní API jako v uvedeném 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` ve volání getJDBCConnection je databázově specifická implementace rozhraní XADataSource , která definuje podrobnosti databáze, ke které se chcete připojit. V dokumentaci k databázi zjistěte, jak vytvořit příslušný objekt XADataSource , který se má předat do getJDBCConnection.

Musíte také aktualizovat cestu ke třídám pomocí příslušných souborů JAR specifických pro databázi, abyste mohli provádět práci s rozhraním JDBC .

Pokud se musíte připojit k více databázím, musíte několikrát volat funkci getJDBCConnection , abyste provedli transakci v několika různých připojeních.

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 ve svých klauzulích throws výjimku, aby se vyhnuly problémům s ověřovatelem prostředí JVM pro zákazníky, kteří nepoužívají funkce JTA. Skutečná vyvolaná výjimka je javax.transaction.xa.XAException , která vyžaduje přidání souboru jta.jar do cesty ke třídám pro programy, které ji 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 problémy a omezení podpory JTA/JDBC závisí na používaném systému správy databází, například testované ovladače JDBC se chovají odlišně, když je databáze vypnuta, když je spuštěna aplikace. Pokud je připojení k databázi, kterou aplikace používá, přerušeno, existují kroky, které může aplikace provést

k opětovnému vytvoření nového připojení ke správci front a k databázi, aby pak mohla tato nová připojení použít k provedení požadované transakční práce.

Protože podpora JTA/JDBC volá ovladače JDBC , může mít implementace těchto ovladačů JDBC významný vliv na chování systému. Testované ovladače JDBC se zejména chovají odlišně, když je databáze vypnuta za běhu aplikace.

Důležité: Vždy se vyhněte náhlému ukončení práce databáze, když existují aplikace, které k ní zadržují otevřená připojení.

Poznámka: Aplikace IBM MQ classes for Java se musí připojit pomocí režimu vazeb, aby produkt IBM MQ fungoval jako koordinátor databáze.

Více sekcí XAResourceManager
Použití více než jedné sekce XAResourceManager v konfiguračním souboru správce front `qm.ini`není podporováno. Jakákoli jiná sekce XAResourceManager než první je ignorována.

Db2

Někdy funkce 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 k produktu Db2 .

Sekce XAResourceManager musí být nakonfigurována pro použití ThreadOfControl=PROCESS. Pro systém Db2 8.1 a vyšší se tato hodnota neshoduje s výchozím podprocesem nastavení řízení pro systém Db2, proto musí být v otevřeném řetězci XA uvedeno toc=p. Příklad sekce 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
```

To nebrání aplikacím systému Java , které používají koordinaci JTA/JDBC , aby samy používaly podprocesy.

Oracle
Volání metody JDBC Connection.close() po funkci MQQueueManager.disconnect () vygeneruje výjimku SQLException. Buď volejte Connection.close() před MQQueueManager.disconnect (), nebo vynechte volání Connection.close().

Zpracování problémů s databázovými připojeními

Když aplikace IBM MQ classes for Java používá podporu JTA/JDBC poskytovanou produktem IBM MQ, provádí obvykle následující kroky:

1. Vytvoří nový objekt MQQueueManager představující připojení ke správci front, který bude jednat 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) předávající ve zdroji XADataSource , který byl vytvořen dříve. To způsobí, že IBM MQ classes for Java vytvoří připojení k databázi.
4. Volá metodu MQQueueManager.begin () ke spuštění transakce XA.
5. Provádí zasílání zpráv a práci s databází.
6. Po dokončení veškeré požadované práce volá metodu MQQueueManager.commit (). Tímto se dokončí transakce XA.

7. Je-li v tomto bodě vyžadována nová transakce XA, může aplikace opakovat kroky 4, 5 a 6.
8. Po dokončení aplikace by měla zavřít databázové připojení vytvořené v kroku 3 a poté volat metodu MQQueueManager.disconnect () pro odpojení od správce front.

IBM MQ classes for Java udržuje interní seznam všech databázových připojení, 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 souboru IBM MQ classes for Javaa předává podrobné informace o volání XA, které je třeba předat do databáze.
2. Produkt IBM MQ classes for Java poté vyhledá příslušné připojení v seznamu a poté toto připojení použije k toku volání XA do databáze.

Dojde-li během tohoto zpracování ke ztrátě připojení k databázi, měla by aplikace:

1. Vraťte zpět jakoukoli existující práci, která byla provedena v rámci transakce, voláním metody MQQueueManager.backout ().
2. Zavřete databázové připojení. To by mělo způsobit, že IBM MQ classes for Java odebere podrobnosti přerušeného databázového připojení z interního 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 databázové připojení voláním metody MQQueueManager.getJDBCConnection(XADataSource).
6. Proveďte transakci znovu.

To umožňuje aplikaci znovu vytvořit nové připojení ke správci front a k databázi a poté tato připojení použít k provedení požadované transakční práce.

Podpora protokolu TLS (Transport Layer Security) v produktu IBM MQ classes for Java

Klientské aplikace IBM MQ classes for Java podporují šifrování TLS. K použití šifrování TLS je vyžadován poskytovatel JSSE.

Klientské aplikace IBM MQ classes for Java používající volbu TRANSPORT (CLIENT) podporují šifrování TLS. Protokol TLS poskytuje šifrování komunikace, ověřování a integritu zpráv. Obvykle se používá k zabezpečení komunikace mezi libovolnými dvěma rovnocennými partnery na Internetu nebo v rámci intranetu.

Produkt IBM MQ classes for Java používá k obsluze šifrování TLS rozšíření JSSE (Java Secure Socket Extension), a proto vyžaduje poskytovatele JSSE. Prostředí JVM prostředí JSE v1.4 mají vestavěného poskytovatele prostředí JSSE. Podrobnosti o tom, jak spravovat a ukládat certifikáty, se mohou u jednotlivých poskytovatelů lišit. Další informace naleznete v dokumentaci poskytovatele JSSE.

Tento oddíl předpokládá, že váš poskytovatel JSSE je správně nainstalován a nakonfigurován a že byly nainstalovány vhodné certifikáty a zpřístupněny vašemu poskytovateli JSSE.

Pokud aplikace klienta IBM MQ classes for Java používá tabulku CCDT (Client Channel Definition Table) pro připojení ke správci front, postupujte podle části “Použití tabulky definic kanálů klienta s volbou IBM MQ classes for Java” na stránce 364.

Povolení TLS v produktu IBM MQ classes for Java

Chcete-li povolit TLS, zadejte CipherSuite. Existují dva způsoby určení CipherSuite.

Protokol TLS je podporován pouze pro připojení klienta. Chcete-li povolit protokol TLS, musíte zadat sadu CipherSuite , která se má použít při komunikaci se správcem front, a tato sada CipherSuite se musí shodovat se sadou CipherSpec nastavenou na cílovém kanálu. Kromě toho musí být pojmenovaná CipherSuite podporována vaším poskytovatelem JSSE. CipherSuites se však liší od specifikací CipherSpecs , a proto mají různé názvy. “TLS CipherSpecs a CipherSuites v adresáři IBM MQ classes for Java” na stránce 396 obsahuje tabulku mapující CipherSpecs podporované produktem IBM MQ na ekvivalentní CipherSuites známé prostředí JSSE.

Chcete-li povolit protokol TLS, zadejte sadu CipherSuite pomocí statické členské proměnné sady sslCipherprostředí MQEnvironment. Následující příklad se připojí ke kanálu SVRCONN s názvem SECURE.SVRCONN.CHANNEL, který byl nastaven tak, aby vyžadoval TLS s CipherSpec TLS_RSA_WITH_AES_128_CBC_SHA256:

```
MQEnvironment.hostname       = "your_hostname";
MQEnvironment.channel        = "SECURE.SVRCONN.CHANNEL";
MQEnvironment.sslCipherSuite = "SSL_RSA_WITH_AES_128_CBC_SHA256";
MQQueueManager qmgr = new MQQueueManager("your_Q_manager");
```

Ačkoli má kanál CipherSpec TLS_RSA_WITH_AES_128_CBC_SHA256, aplikace Java musí uvést CipherSuite parametru SSL_RSA_WITH_AES_128_CBC_SHA256. Seznam mapování mezi specifikacemi CipherSpecs a CipherSuitesnaleznete v části “TLS CipherSpecs a CipherSuites v adresáři IBM MQ classes for Java” na stránce 396 .

Aplikace může také určit CipherSuite nastavením vlastnosti prostředí CMQC.SSL_CIPHER_SUITE_PROPERTY.

Případně použijte tabulku CCDT (Client Channel Definition Table). Další informace viz “Použití tabulky definic kanálů klienta s volbou IBM MQ classes for Java” na stránce 364

Pokud požadujete, aby připojení klienta používalo sadu CipherSuite , kterou podporuje poskytovatel IBM Java JSSE FIPS (IBMJSSEFIPS), může aplikace nastavit pole sslFipsRequired ve třídě MQEnvironment na hodnotu `true`. Alternativně může aplikace nastavit vlastnost prostředí CMQC.SSL_FIPS_REQUIRED_PROPERTY. Výchozí hodnota je `false`, což znamená, že připojení klienta může používat libovolnou sadu CipherSuite podporovanou produktem IBM MQ.

Pokud aplikace používá více než jedno připojení klienta, hodnota pole sslFipsNezbytné pole, které 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 pole sslFipsRequired se ignoruje. Chcete-li pro pole sslFipspoužít jinou hodnotu, musíte restartovat aplikaci.

Chcete-li se úspěšně připojit pomocí TLS, musí být úložiště údajů o důvěryhodnosti JSSE nastaveno pomocí kořenových certifikátů certifikační autority, z nichž lze ověřit certifikát předložený správcem front. Podobně, pokud byl parametr SSLClientAuth v kanálu SVRCONN nastaven na hodnotu MQSSL_CLIENT_AUTH_REQUIRED, musí úložiště klíčů JSSE obsahovat identifikační certifikát, který je důvěryhodný pro správce front.

Související odkazy

Standard FIPS (Federal Information Processing Standards) pro AIX, Linux, and Windows

Použití rozlišujícího názvu správce front v adresáři IBM MQ classes for Java

Správce front se identifikuje pomocí certifikátu TLS, který obsahuje rozlišující název (DN). Aplikace klienta IBM MQ classes for Java může toto DN používat, aby se ujistila, že komunikuje se správným správcem front.

Vzor DN je uveden pomocí proměnné názvu sslPeerprostředí MQEnvironment. Například nastavení:

```
MQEnvironment.sslPeerName = "CN=QMGR.*, OU=IBM, OU=WEBSPHERE";
```

umožňuje úspěšné připojení pouze v případě, že správce front předloží certifikát s obecným názvem začínajícím na QMGR., a alespoň dva názvy organizačních jednotek, z nichž první musí být IBM a druhý WebSphere.

Je-li nastaven název sslPeer, připojení budou úspěšná pouze v případě, že je nastaven na platný vzor a správce front předloží odpovídající certifikát.

Aplikace může také zadat rozlišující název správce front nastavením vlastnosti prostředí CMQC.SSL_PEER_NAME_PROPERTY. Další informace o rozlišujících názvech naleznete v tématu Rozlišovací názvy.

Použití seznamů odvolaných certifikátů v adresáři 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 vůči uvedenému seznamu CRL.

Seznam odvolaných certifikátů (CRL) je sada certifikátů, které byly odvolány vydávající certifikační autoritou nebo místní organizací. Seznamy CRL jsou obvykle hostovány na serverech LDAP. S verzí Java 2 v1.4lze určit server CRL v době připojení a certifikát předložený správcem front je před povolením připojení zkontrolován proti seznamu CRL. Další informace o seznamech odvolaných certifikátů a IBM MQnaleznete v tématu Práce se seznamy odvolaných certifikátů a seznamy odvolaných oprávnění a Přístup k seznamům CRL a ARL pomocí IBM MQ classes for Java a IBM MQ classes for JMS.

Poznámka: Chcete-li úspěšně použít CertStore se seznamem CRL hostovaným na serveru LDAP, ujistěte se, že je vaše sada Java Software Development Kit (SDK) kompatibilní se seznamem CRL. Některé sady SDK vyžadují, aby seznam CRL odpovídal standardu RFC 2587, který definuje schéma pro protokol LDAP v2. Většina serverů LDAP v3 místo toho používá RFC 2256.

Seznamy CRL, které se mají použít, jsou určeny prostřednictvím třídy java.security.cert.CertStore . Úplné podrobnosti o tom, jak získat instance úložiště CertStore, naleznete v dokumentaci k této třídě. Chcete-li vytvořit úložiště CertStore založené na serveru LDAP, nejprve vytvořte instanci parametrů LDAPCertStore, inicializovanou s použitím nastavení serveru a portu. Příklad:

```
import java.security.cert.*;
CertStoreParameters csp = new LDAPCertStoreParameters("crl_server", 389);
```

Po vytvoření instance CertStoreParameters použijte statický konstruktor na CertStore k vytvoření CertStore typu LDAP:

```
CertStore cs = CertStore.getInstance("LDAP", csp);
```

Podporovány jsou i jiné typy CertStore (například Collection). Obvykle existuje několik serverů CRL nastavených s identickými informacemi CRL, které poskytují redundanci. Máte-li objekt CertStore pro každý z těchto serverů CRL, umístěte je všechny do vhodné kolekce. Následující příklad ukazuje objekty CertStore umístěné v seznamu ArrayList:

```
import java.util.ArrayList;
Collection crls = new ArrayList();
crls.add(cs);
```

Tuto kolekci lze nastavit do statické proměnné MQEnvironment sslCertStores před připojením k povolení kontroly CRL:

```
MQEnvironment.sslCertStores = crls;
```

Certifikát předložený správcem front při nastavování připojení je ověřen následujícím způsobem:

1. První objekt CertStore v kolekci určené úložištěm sslCertse používá k identifikaci serveru CRL.
2. Došlo k pokusu o kontaktování serveru CRL.
3. Pokud je pokus úspěšný, server vyhledá shodu pro certifikát.
   a. Pokud se zjistí, že je certifikát odvolán, proces vyhledávání je ukončen a požadavek na připojení se nezdaří s kódem příčiny MQRC_SSL_CERTIFICATE_ODVOLÁN.
   b. Není-li certifikát nalezen, proces vyhledávání je ukončen a připojení může pokračovat.
4. Pokud je pokus o kontaktování serveru neúspěšný, další objekt CertStore se použije k identifikaci serveru CRL a proces se opakuje od kroku 2.

   Pokud se jednalo o poslední CertStore v kolekci nebo pokud kolekce neobsahuje žádné objekty CertStore , proces vyhledávání se nezdařil a požadavek na připojení se nezdaří s kódem příčiny MQRC_SSL_CERT_STORE_ERROR.

Objekt Collection určuje pořadí, ve kterém se používají úložiště CertStores .

Kolekci CertStores lze také nastavit pomocí CMQC.SSL_CERT_STORE_PROPERTY. Pro usnadnění umožňuje tato vlastnost také zadat jeden objekt CertStore , aniž by byl členem kolekce.

Je-li pro úložiště sslCertnastavena hodnota null, neprovádí se žádná kontrola CRL. Tato vlastnost je ignorována, pokud není nastavena sada sslCipher.

Opětovné vyjednání tajného klíče v souboru IBM MQ classes for Java
Klientská aplikace IBM MQ classes for Java může řídit, kdy je znovu vyjednán tajný klíč, který se používá pro šifrování na připojení klienta, ve smyslu 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 pořadí.

- Nastavením pole Počet sslResetve třídě MQEnvironment.
- Nastavením vlastnosti prostředí MQC.SSL_RESET_COUNT_PROPERTY v objektu hašovací tabulky. Aplikace poté přiřadí hašovací tabulku k poli `properties` ve třídě MQEnvironment nebo předá hašovací tabulku objektu MQQueueManager v konstruktoru.

Hodnota pole sslResetPočet nebo vlastnost prostředí MQC.SSL_RESET_COUNT_PROPERTY představuje celkový počet bajtů odeslaných a přijatých kódem klienta IBM MQ classes for Java před opětovným vyjednáním tajného klíče. Počet odeslaných bajtů je číslo před šifrováním a počet přijatých bajtů je číslo po dešifrování. Počet bajtů také zahrnuje řídicí informace odeslané a přijaté klientem IBM MQ classes for Java .

Pokud je počet resetů nula, což je výchozí hodnota, tajný klíč nebude nikdy znovu vyjednán. Není-li zadána žádná sada CipherSuite , bude počet resetů ignorován.

Dodání přizpůsobeného SSLSocketFactory v IBM MQ classes for Java
Používáte-li přizpůsobenou továrnu soketů JSSE, nastavte MQEnvironment.sslSocketFactory na přizpůsobený objekt továrny. 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 úložišť klíčů a úložišť údajů o důvěryhodnosti podle programů nebo umožňují změnu výběru certifikátu identity z úložiště klíčů. V prostředí JSSE jsou všechny tyto úpravy abstrahovány do třídy továrny javax.net.ssl.SSLSocketFactory.

Podrobné informace o tom, jak vytvořit přizpůsobenou implementaci SSLSocketFactory , naleznete v dokumentaci JSSE. Podrobnosti se liší od poskytovatele k poskytovateli, ale typická posloupnost kroků může být:

1. Vytvořit objekt SSLContext pomocí statické metody na SSLContext
2. Inicializujte tento SSLContext s příslušnými implementacemi KeyManager a TrustManager (vytvořenými z jejich vlastních tříd továrny).
3. Vytvořte SSLSocketFactory z SSLContext.

Máte-li objekt SSLSocketFactory , nastavte vlastnost MQEnvironment.sslSocketFactory na přizpůsobený objekt továrny. Příklad:

```
javax.net.ssl.SSLSocketFactory sf = sslContext.getSocketFactory();
MQEnvironment.sslSocketFactory = sf;
```

IBM MQ classes for Java použijte tento SSLSocketFactory pro připojení ke správci front IBM MQ .
Tuto vlastnost lze nastavit také pomocí CMQC.SSL_SOCKET_FACTORY_PROPERTY. Pokud je továrna sslSocketnastavena na hodnotu null, použije se výchozí hodnota SSLSocketFactory prostředí JVM. Tato vlastnost je ignorována, pokud není nastavena sada sslCipher.

Při použití vlastního parametru SSLSocketFactorieszvažte vliv sdílení připojení TCP/IP. Je-li možné sdílení připojení, není požadován nový soket od dodané továrny SSLSocketFactory , a to ani v případě, že by

se vytvořený soket nějakým způsobem lišil v kontextu následného požadavku na připojení. Má-li být například při následném připojení předložen jiný certifikát klienta, nesmí být sdílení připojení povoleno.

Provedení změn v úložišti klíčů nebo úložišti údajů o důvěryhodnosti JSSE v IBM MQ classes for Java
Pokud změníte úložiště klíčů JSSE nebo úložiště údajů o důvěryhodnosti, musíte provést určité akce, aby se změny projevily.

Pokud změníte obsah úložiště klíčů nebo úložiště údajů o důvěryhodnosti JSSE nebo změníte umístění souboru úložiště klíčů nebo úložiště údajů o důvěryhodnosti, aplikace IBM MQ classes for Java , které jsou spuštěny v daném okamžiku, změny automaticky nevyzvednou. Aby se změny projevily, musí být provedeny následující akce:

- Aplikace musí zavřít všechna svá připojení a zničit všechna 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 tom, jak navrhujete aplikace, a na funkci, kterou poskytuje poskytovatel JSSE, může být možné provést tyto akce bez zastavení a restartování aplikací. Nejjednodušším řešením však může být zastavení a restartování aplikací.

Ošetření chyb při použití TLS s IBM MQ classes for Java
Produkt IBM MQ classes for Java může při připojování ke správci front pomocí protokolu TLS vydat řadu kódů příčiny.

Tyto informace jsou vysvětleny v následujícím seznamu:

MQRC_SSL_NOT_ALLOWED
Vlastnost sslCipherSuite byla nastavena, ale bylo použito připojení vazeb. TLS podporuje pouze připojení klienta.

MQRC_JSSE_ERROR
Poskytovatel JSSE ohlásil chybu, kterou nelze zpracovat pomocí IBM MQ. Příčinou může být problém s konfigurací prostředí JSSE nebo skutečnost, že certifikát předložený správcem front nelze ověřit. Výjimku vytvořenou rozhraním JSSE lze načíst pomocí metody getCause() na základě výjimky MQException.

MQRC_SSL_INITIALIZATION_ERROR
Bylo zadáno volání MQCONN nebo MQCONNX se zadanými volbami konfigurace TLS, ale během inicializace prostředí TLS došlo k chybě.

MQRC_SSL_PEER_NAME_MISMATCH
Vzor DN uvedený ve vlastnosti názvu sslPeerse neshodoval s rozlišujícím názvem předloženým správcem front.

CHYBA-MQRC_SSL_PEER_NAME_ERROR
Vzor DN uvedený ve vlastnosti názvu sslPeernebyl platný.

MQRC_UNSUPPORTED_CIPHER_SUITE
CipherSuite pojmenovaná v sadě sslCiphernebyla poskytovatelem JSSE rozpoznána. Úplný seznam CipherSuites podporovaných poskytovatelem JSSE lze získat pomocí programu pomocí metody SSLSocketFactory.getSupportedCipherSuites(). Seznam CipherSuites , které lze použít ke komunikaci s produktem IBM MQ , naleznete v adresáři “TLS CipherSpecs a CipherSuites v adresáři IBM MQ classes for Java” na stránce 396.

MQRC_SSL_CERTIFICATE_ODVOLÁNO
Certifikát předložený správcem front byl nalezen v seznamu CRL určeném pomocí vlastnosti sslCertStores. Aktualizujte správce front tak, aby používal důvěryhodné certifikáty.

MQRC_SSL_CERT_STORE_ERROR
V žádném z dodaných úložišť CertStores nebylo možné vyhledat certifikát předložený správcem front. Metoda MQException.getCause() vrací chybu, ke které došlo při hledání prvního pokusu o operaci CertStore . Pokud je příčinou výjimka NoSuchElementException, ClassCastnebo výjimka NullPointer,

zkontrolujte, zda kolekce určená ve vlastnosti sslCertStores obsahuje alespoň jeden platný objekt CertStore .

TLS CipherSpecs a CipherSuites v adresáři IBM MQ classes for Java
Schopnost aplikací IBM MQ classes for Java navázat připojení ke správci front závisí na specifikaci CipherSpec zadané na konci serveru kanálu MQI a na sadě CipherSuite zadané na konci klienta.

V následující tabulce jsou uvedeny specifikace CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites.

Deprecated Měli byste si prostudovat téma Zamítnuté CipherSpecs , abyste zjistili, zda některá ze specifikací CipherSpecsuvedených v následující tabulce nebyla zamítnuta produktem IBM MQ , a pokud ano, při které aktualizaci specifikace CipherSpec byla zamítnuta.

Důležité: V seznamu CipherSuites jsou uvedeny ty, které jsou podporovány prostředím JRE (IBM Java Runtime Environment) dodávaným s produktem IBM MQ. Mezi uvedené CipherSuites patří ty, které jsou podporovány prostředím Oracle Java JRE. Další informace o konfiguraci aplikace pro použití prostředí Oracle Java JRE viz “Konfigurace aplikace pro použití mapování produktu IBM Java nebo Oracle Java CipherSuite” na stránce 417.

Tabulka také označuje protokol, který se používá pro komunikaci, a zda je či není sada CipherSuite v souladu se standardem FIPS 140-2.

Poznámka: V systému AIX, Linux, and Windowsposkytuje produkt IBM MQ kompatibilitu se standardem FIPS 140-2 prostřednictvím šifrovacího modulu IBM Crypto for C (ICC) . Certifikát pro tento modul byl přesunut do historického stavu. Zákazníci by si měli prohlédnout IBM Crypto for C (ICC) certifikát a měli by si být vědomi všech doporučení poskytnutých NIST. Náhradní modul FIPS 140-3 momentálně probíhá a jeho stav lze zobrazit jeho vyhledáním v modulech NIST CMVP v seznamu procesů.

Šifrovací sady označené jako vyhovující FIPS 140-2 lze použít, pokud nebyla aplikace nakonfigurována tak, aby vynucovala kompatibilitu se standardem FIPS 140-2, ale pokud byla pro aplikaci konfigurována shoda se standardem FIPS 140-2 (viz následující poznámky o konfiguraci), lze konfigurovat pouze ty CipherSuites , které jsou označeny jako kompatibilní se standardem FIPS 140-2; při pokusu o použití jiných CipherSuites dojde k chybě.

Poznámka: Každé prostředí JRE může mít více poskytovatelů kryptografického zabezpečení, z nichž každý může přispívat implementací stejné CipherSuite. Avšak ne všichni poskytovatelé zabezpečení mají certifikaci FIPS 140-2. Není-li pro aplikaci vynucena shoda se standardem FIPS 140-2, je možné, že bude použita necertifikovaná implementace CipherSuite . Necertifikované implementace nemusí fungovat v souladu se standardem FIPS 140-2, i když sada CipherSuite teoreticky splňuje minimální úroveň zabezpečení vyžadovanou standardem. Další informace o konfiguraci vynucení FIPS 140-2 v aplikacích IBM MQ Java naleznete v následujících poznámkách.

Další informace o kompatibilních standardech FIPS 140-2 a Suite-B pro CipherSpecs a CipherSuitesnaleznete v tématu Určení CipherSpecs. Možná budete muset mít na paměti informace, které se týkají Federálních standardů zpracování informací.

Chcete-li použít úplnou sadu CipherSuites a pracovat s certifikovaným standardem FIPS 140-2 a/nebo Suite-B, je vyžadováno vhodné prostředí JRE. IBM Java 7 Service Refresh 4 Fix Pack 2 nebo vyšší úroveň prostředí IBM JRE poskytuje odpovídající podporu pro TLS 1.2 CipherSuites uvedené v části Tabulka 59 na stránce 397.

Chcete-li používat šifry TLS 1.3 , musí prostředí JRE spouštějící vaši aplikaci podporovat protokol TLS 1.3.

Poznámka: Chcete-li použít některé CipherSuites, je třeba v prostředí JRE nakonfigurovat soubory zásad 'bez omezení'. Další podrobnosti o nastavení souborů zásad v sadě SDK nebo JRE naleznete v tématu *IBM Soubory zásad sady SDK* v příručce *Security Reference for IBM SDK, Java Technology Edition* pro používanou verzi.

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites

CipherSpec **"1" na stránce 416**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem 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	TLS 1.2	yes

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites (pokračování)

CipherSpec **"1" na stránce 416**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem 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	TLS 1.2	yes

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites (pokračování)

CipherSpec **"1" na stránce 416**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem 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	TLS 1.2	yes

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites (pokračování)

CipherSpec **"1" na stránce 416**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem 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	TLS 1.2	yes

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites (pokračování)

CipherSpec **"1" na stránce 416**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem 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	TLS 1.2	yes

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites (pokračování)

CipherSpec **"1" na stránce 416**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem FIPS 140-2**
ECDHE_ECDSA_NULL_SHA256	SSL_ECDHE_ECDSA_WITH_NULL_SHA	TLS_ECDHE_ECDSA_WITH_NULL_SHA	TLS 1.2	no
ECDHE_ECDSA_RC4_128_SHA256	SSL_ECDHE_ECDSA_WITH_RC4_128_SHA	TLS_ECDHE_ECDSA_WITH_RC4_128_SHA	TLS 1.2	no

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites (pokračování)

CipherSpec **"1" na stránce 416**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem 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	TLS 1.2	yes

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites (pokračování)

CipherSpec **"1" na stránce 416**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem 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	TLS 1.2	yes

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites (pokračování)

CipherSpec **"1" na stránce 416**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem 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	TLS 1.2	yes

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites (pokračování)

CipherSpec **"1" na stránce 416**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem 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	TLS 1.2	yes

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites (pokračování)

CipherSpec **"1" na stránce 416**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem 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	TLS 1.2	yes
ECDHE_RSA_NULL_SHA256	SSL_ECDHE_RSA_WITH_NULL_SHA	TLS_ECDHE_RSA_WITH_NULL_SHA	TLS 1.2	no

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites (pokračování)

CipherSpec **"1" na stránce 416**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem FIPS 140-2**
ECDHE_RSA_RC4_128_SHA256	SSL_ECDHE_RSA_WITH_RC4_128_SHA	TLS_ECDHE_RSA_WITH_RC4_128_SHA	TLS 1.2	no
TLS_RSA_WITH_3DES_EDE_CBC_SHA "2" na stránce 416	SSL_RSA_WITH_3DES_EDE_CBC_SHA	TLS_RSA_WITH_3DES_EDE_CBC_SHA	TLS 1.0	ne "4" na stránce 416

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites (pokračování)

CipherSpec **“1” na stránce 416**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem FIPS 140-2**
TLS_RSA_WITH_AES_128_CBC_SHA	SSL_RSA_WITH_AES_128_CBC_SHA	TLS_RSA_WITH_AES_128_CBC_SHA	TLS 1.0	ne “4” na stránce 416
TLS_RSA_WITH_AES_128_CBC_SHA256	SSL_RSA_WITH_AES_128_CBC_SHA256	TLS_RSA_WITH_AES_128_CBC_SHA256	TLS 1.2	ne “4” na stránce 416

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites (pokračování)

CipherSpec **"1" na stránce 416**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem FIPS 140-2**
TLS_RSA_WITH_AES_128_GCM_SHA256	SSL_RSA_WITH_AES_128_GCM_SHA256	TLS_RSA_WITH_AES_128_GCM_SHA256	TLS 1.2	ne "4" na stránce 416
TLS_RSA_WITH_AES_256_CBC_SHA	SSL_RSA_WITH_AES_256_CBC_SHA	TLS_RSA_WITH_AES_256_CBC_SHA	TLS 1.0	ne "4" na stránce 416

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites (pokračování)

CipherSpec **"1" na stránce 416**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem FIPS 140-2**
TLS_RSA_WITH_AES_256_CBC_SHA256	SSL_RSA_WITH_AES_256_CBC_SHA256	TLS_RSA_WITH_AES_256_CBC_SHA256	TLS 1.2	ne "4" na stránce 416
TLS_RSA_WITH_AES_256_GCM_SHA384	SSL_RSA_WITH_AES_256_GCM_SHA384	TLS_RSA_WITH_AES_256_GCM_SHA384	TLS 1.2	ne "4" na stránce 416

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites (pokračování)

CipherSpec **"1" na stránce 416**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem FIPS 140-2**
TLS_RSA_WITH_DES_CBC_SHA	SSL_RSA_WITH_DES_CBC_SHA	SSL_RSA_WITH_DES_CBC_SHA	TLS 1.0	no
TLS_RSA_WITH_NULL_SHA256	SSL_RSA_WITH_NULL_SHA256	TLS_RSA_WITH_NULL_SHA256	TLS 1.2	no

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites (pokračování)

CipherSpec **"1" na stránce 416**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem FIPS 140-2**
TLS_RSA_WITH_RC4_128_SHA256	SSL_RSA_WITH_RC4_128_SHA	SSL_RSA_WITH_RC4_128_SHA	TLS 1.2	no
ANY_TLS12	*TLS12	*TLS12	TLS 1.2	yes
TLS_AES_128_GCM_SHA256 "3" na stránce 416	TLS_AES_128_GCM_SHA256	TLS_AES_128_GCM_SHA256	TLS v1.3	no

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites (pokračování)

CipherSpec **"1" na stránce 416**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem FIPS 140-2**
TLS_AES_256_GCM_SHA384 "3" na stránce 416	TLS_AES_256_GCM_SHA384	TLS_AES_256_GCM_SHA384	TLS v1.3	no
TLS_CHACHA20_POLY1305_SHA256 "3" na stránce 416	TLS_CHACHA20_POLY1305_SHA256	TLS_CHACHA20_POLY1305_SHA256	TLS v1.3	no

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites (pokračování)

CipherSpec **"1" na stránce 416**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem FIPS 140-2**
TLS_AES_128_CCM_SHA256 "3" na stránce 416	TLS_AES_128_CCM_SHA256	TLS_AES_128_CCM_SHA256	TLS v1.3	no
TLS_AES_128_CCM_8_SHA256 "3" na stránce 416	TLS_AES_128_CCM_8_SHA256	TLS_AES_128_CCM_8_SHA256	TLS v1.3	no
Libovolný "3" na stránce 416	*ANY	*ANY	Více	no
ANY_TLS13 "3" na stránce 416	*TLS13	*TLS13	TLS V13	no

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalentní CipherSuites (pokračování)				
CipherSpec **"1" na stránce 416**	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní se standardem FIPS 140-2**
ANY_TLS12_OR_HIGHER "3" na stránce 416	*TLS12ORHIGHER	*TLS12ORHIGHER	TLS 1.2 a vyšší	no
ANY_TLS13_OR_HIGHER "3" na stránce 416	*TLS13ORHIGHER	*TLS13ORHIGHER	TLS 1.3 a vyšší	no

Notes:

1. Toto je hodnota nakonfigurovaná na kanálu v produktu IBM MQ, včetně CCDT (binární nebo JSON).
2. Deprecated CipherSpec TLS_RSA_WITH_3DES_EDE_CBC_SHA je zamítnuta. Přesto jej lze použít k přenosu až 32 GB dat před ukončením připojení s chybou AMQ9288. Chcete-li se vyhnout této chybě, musíte se buď vyhnout použití trojitého DES, nebo povolit reset tajného klíče při použití této CipherSpec.
3. Chcete-li používat šifry TLS v1.3 , musí prostředí Java runtime environment (JRE), které spouští vaši aplikaci, podporovat protokol TLS v1.3.
4. V 9.3.0.17 V 9.3.5.1 V systémech IBM MQ 9.3.5 CSU 1 a IBM MQ 9.3.0 CSU 17prostředí JRE produktu IBM Java 8 odebere podporu výměny klíčů RSA při práci v režimu FIPS.

Konfigurace šifrovacích sad a shody se standardem FIPS v aplikaci IBM MQ classes for Java

- Aplikace, která používá produkt IBM MQ classes for Java , může k nastavení CipherSuite pro připojení použít jednu ze dvou metod:

  - Nastavte pole sslCipherSuite ve třídě MQEnvironment na název CipherSuite .
  - Nastavte vlastnost CMQC.SSL_CIPHER_SUITE_PROPERTY v hašovací tabulce vlastností předané konstruktoru MQQueueManager na název CipherSuite .
- Aplikace, která používá produkt IBM MQ classes for Java , může k vynucení shody se standardem FIPS 140-2 použít jednu ze dvou metod:
  - Nastavte pole sslFipsve třídě MQEnvironment na hodnotu true.
  - Nastavte vlastnost CMQC.SSL_FIPS_REQUIRED_PROPERTYin hašovací tabulku vlastností předanou konstruktoru MQQueueManager na hodnotu true.

Konfigurace aplikace pro použití mapování produktu IBM Java nebo Oracle Java CipherSuite

Poznámka:

V 9.3.3 V případě produktu Continuous Delivery from IBM MQ 9.3.3je Java Vlastnost systému `com.ibm.mq.cfg.useIBMCipherMappings`, která řídí, která mapování se používají, odebrána z produktu. V produktu IBM MQ 9.3.3lze šifrovací modul definovat buď jako název CipherSpec , nebo jako název CipherSuite a produkt IBM MQjej správně zpracuje. Následující tři soubory JAR Jackson jsou povinné, pokud vaše aplikace IBM MQ classes for Java vytvoří zabezpečená připojení TLS ke správci front:

- `jackson-annotations.jar`
- `jackson-core.jar`
- `jackson-databind.jar`

Důležité: Následující informace o souboru `com.ibm.mq.cfg.useIBMCipherMappings` platí pouze pro položky Long Term Support a Continuous Delivery před IBM MQ 9.3.3 .

Můžete nakonfigurovat, zda vaše aplikace používá výchozí mapování IBM Java CipherSuite na IBM MQ CipherSpec , nebo mapování Oracle CipherSuite na IBM MQ CipherSpec . Proto můžete použít TLS CipherSuites bez ohledu na to, zda vaše aplikace používá prostředí IBM JRE nebo Oracle JRE. Java Vlastnost systému `com.ibm.mq.cfg.useIBMCipherMappings` řídí, která mapování se použijí. Vlastnost může mít jednu z následujících hodnot:

ano
Použijte mapování IBM Java CipherSuite na IBM MQ CipherSpec .

Tato hodnota je výchozí hodnota.

ne
Použijte mapování Oracle CipherSuite na IBM MQ CipherSpec .

Další informace o použití šifer IBM MQ Java a TLS viz příspěvek blogu MQdev MQ Java, šifry TLS, jiné nežIBM JRE a APAR IT06775, IV66840, IT09423, IT10837.

Omezení interoperability

Určité CipherSuites mohou být kompatibilní s více než jednou IBM MQ CipherSpec, v závislosti na používaném protokolu. Podporována je však pouze kombinace CipherSuite/CipherSpec , která používá verzi TLS uvedenou v tabulce 1. Při pokusu o použití nepodporovaných kombinací CipherSuites a CipherSpecs dojde k selhání s příslušnou výjimkou. Instalace používající libovolnou z těchto kombinací CipherSuite/CipherSpec by se měly přesunout na podporovanou kombinaci.

Následující tabulka zobrazuje CipherSuites , na které se toto omezení vztahuje.

Tabulka 60. CipherSuites a jejich podporované a nepodporované CipherSpecs		
CipherSuite	**Podporovaná specifikace TLS CipherSpec**	**Nepodporovaná specifikace SSL CipherSpec**
SSL_RSA_WITH_3DES_EDE_CBC_SHA	TLS_RSA_WITH_3DES_EDE_CBC_SHA "1" na stránce 418	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. Deprecated Tato CipherSpec TLS_RSA_WITH_3DES_EDE_CBC_SHA je zamítnutá. Přesto jej lze použít k přenosu až 32 GB dat před ukončením připojení s chybou AMQ9288. Chcete-li se vyhnout této chybě, musíte se buď vyhnout použití trojitého DES, nebo povolit reset tajného klíče při použití této CipherSpec.

Spuštění aplikací IBM MQ classes for Java

Pokud napíšete aplikaci (třídu, která obsahuje metodu main ()), pomocí režimu klienta nebo režimu vazeb, spusťte program pomocí interpretu Java .

Použijte příkaz:

```
java -Djava.library.path= library_path MyClass
```

kde *cesta_k_knihovně* je cesta ke knihovnám IBM MQ classes for Java . Další informace viz "IBM MQ classes for Java knihovny" na stránce 345.

Související úlohy

Trasování aplikací IBM MQ classes for Java

Trasování adaptéru prostředků IBM MQ

Chování závislé na prostředí IBM MQ classes for Java

Produkt IBM MQ classes for Java vám umožňuje vytvářet aplikace, které lze spouštět pro různé verze produktu IBM MQ. Tato kolekce témat popisuje chování tříd Java závislých na těchto různých verzích.

Produkt IBM MQ classes for Java poskytuje jádro tříd, které poskytují konzistentní funkce a chování ve všech prostředích. Funkce mimo toto jádro závisí na schopnosti správce front, ke kterému je aplikace připojena.

S výjimkou, kde je zde uvedeno, je vystavené chování takové, jak je popsáno v odkazu na aplikaci MQI odpovídajícím správci front.

Hlavní třídy v souboru IBM MQ classes for Java

IBM MQ classes for Java obsahuje základní sadu tříd, kterou lze použít ve všech prostředích.

Následující sada tříd je považována za hlavní třídy a lze ji použít ve všech prostředích s pouze vedlejšími variacemi uvedenými v části "Omezení a variace pro hlavní třídy produktu IBM MQ classes for Java" na stránce 419.

- Prostředí MQEnvironment
- Výjimka MQException
- Volby MQGetMessage

  Vyloučení:

  – MatchOptions

  - GroupStatus
  - SegmentStatus
  - Segmentace
- MQManagedObject

  Vyloučení:
  - dotazovat ()
  - set () (nastavení)
- Zpráva MQMessage

  Vyloučení:
  - groupId
  - messageFlags
  - Číslo messageSequence
  - posunutí
  - originalLength
- MQPoolServices
- Událost MQPoolServices
- MQPoolServicesEventListener
- MQPoolToken
- Volby MQPutMessage

  Vyloučení:
  - KnownDestCount
  - UnknownDestCount
  - InvalidDestCount
  - recordFields
- MQProcess
- MQQUEUE
- MQQueueManager

  Vyloučení:
  - Počáteční ()
  - accessDistributionList ()
- Správce MQSimpleConnection
- Téma MQTopic
- MQC

Poznámka:

1. Některé konstanty nejsou součástí jádra (podrobnosti viz “Omezení a variace pro hlavní třídy produktu IBM MQ classes for Java” na stránce 419); nepoužívejte je ve zcela přenosných programech.
2. Některé platformy nepodporují všechny režimy připojení. Na těchto platformách můžete použít pouze hlavní třídy a volby, které se vztahují k podporovaným režimům.

Omezení a variace pro hlavní třídy produktu IBM MQ classes for Java

Hlavní třídy se obecně chovají konzistentně ve všech prostředích, a to i v případě, že ekvivalentní volání MQI mají obvykle rozdíly v prostředí. Chování je stejné, jako kdyby byl použit správce front AIX, Linuxnebo Windows , s výjimkou následujících menších omezení a variací.

*Omezení pro hodnoty MQGMO_ * v souboru IBM MQ classes for Java*
Určité hodnoty MQGMO_ * nejsou podporovány všemi správci front.

Použití následujících hodnot MQGMO_ * může vést k vyvolání výjimky MQException z funkce MQQueue.get():

MQGMO_SYNCPOINT_IF_PERSISTENT
MQGMO_MARK_SKIP_BACKOUT
MQGMO_BROWSE_MSG_UNDER_CURSOR
MQGMO_LOCK
MQGMO_UNLOCK
MQGMO_LOGICAL_ORDER
MQGMO_COMPLETE_MESSAGE
MQGMO_ALL_MSGS_AVAILABLE
MQGMO_ALL_SEGMENTS_AVAILABLE
MQGMO_UNMARKED_BROWSE_MSG
MQGMO_MARK_BROWSE_HANDLE
MQGMO_MARK_BROWSE_CO_OP
MQGMO_UNMARK_BROWSE_HANDLE
MQGMO_UNMARK_BROWSE_CO_OP

Kromě toho není MQGMO_SET_SIGNAL při použití z produktu Javapodporován.

*Omezení pro hodnoty MQPMRF_ * v souboru IBM MQ classes for Java*
Používají se pouze při vkládání zpráv do distribučního seznamu a jsou podporovány pouze správci front, kteří podporují distribuční seznamy. Například správci front z/OS nepodporují distribuční seznamy.

*Omezení pro hodnoty MQPMO_ * v souboru IBM MQ classes for Java*
Určité hodnoty MQPMO_ * nejsou podporovány všemi správci front.

Použití následujících hodnot MQPMO_ * může vést k vyvolání výjimky MQException z funkce MQQueue.put() nebo MQQueueManager.put ():

MQPMO_LOGICAL_ORDER
MQPMO_NEW_CORREL_ID
MQPMO_NEW_MESSAGE_ID
MQPMO_RESOLVE_LOCAL_Q

*Omezení a variace pro hodnoty MQCNO_ * v souboru IBM MQ classes for Java*
Určité hodnoty `MQCNO_*` nejsou podporovány.

- Automatické opětovné připojení klienta není produktem IBM MQ classes for Javapodporováno. Bez ohledu na hodnotu `MQCNO_RECONNECT_*` , kterou jste nastavili, se připojení bude i nadále chovat, jako byste nastavili `MQCNO_RECONNECT_DISABLED`.
- Ve správcích front, kteří nepodporují produkt `MQCNO_FASTPATH`, je parametr `MQCNO_FASTPATH` ignorován. Připojení klienta jej také ignoruje.

*Omezení pro hodnoty MQRO_ * v souboru IBM MQ classes for Java*
Lze nastavit 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_EXPIRAČNÍ

Další informace viz Sestava.

Různé rozdíly mezi IBM MQ classes for Java na z/OS a jiných platformách
IBM MQ for z/OS se v některých oblastech chová jinak než IBM MQ na jiných platformách.

BackoutCount
Správce front z/OS vrací maximální hodnotu BackoutCount 255, a to i v případě, že zpráva byla vrácena zpět více než 255 krát.

Výchozí předpona dynamické fronty
Při připojení ke správci front z/OS pomocí připojení vazeb je výchozí předpona dynamické fronty CSQ. *. Jinak je výchozí předpona dynamické fronty AMQ. *.

Konstruktor MQQueueManager
Připojení klienta není v systému z/OSpodporováno. Pokus o připojení s volbami 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 mu nepodaří inicializovat převod mezi kódovými stránkami IBM-1047 a ISO8859-1) nebo MQRC_UCS2_CONVERSION_ERROR (pokud se nezdaří inicializovat převod mezi kódovou stránkou správce front a kódovou stránkou Unicode). Pokud vaše aplikace selže s jedním z těchto kódů příčiny, ujistěte se, že je nainstalována komponenta prostředků národního jazyka jazykového prostředí, a ujistěte se, že jsou k dispozici správné převodní tabulky.

Převodní tabulky pro Unicode se instalují 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
Produkt IBM MQ classes for Java obsahuje určité funkce, které jsou speciálně navrženy tak, aby používaly rozšíření rozhraní API, která nejsou podporována všemi správci front. Tato kolekce témat popisuje, jak se chovají při použití správce front, který je nepodporuje.

Varianty volby konstruktoru MQQueueManager
Některé konstruktory MQQueueManager obsahují volitelný celočíselný argument. Některé hodnoty tohoto argumentu nejsou přijaty na všech platformách.

Pokud konstruktor MQQueueManager obsahuje volitelný celočíselný argument, mapuje se na pole voleb MQI MQCNO a používá se k přepínání mezi normálním a rychlým připojením. Tato rozšířená forma konstruktoru je přijata ve všech prostředích, pokud jsou použity pouze volby MQCNO_STANDARD_BINDING nebo MQCNO_FASTPATH_BINDING. Jakékoli jiné volby způsobí selhání konstruktoru s MQRC_OPTIONS_ERROR. Volba rychlé cesty CMQC.MQCNO_FASTPATH_BINDING je uznána pouze s vazebním připojením ke správci front, který jej podporuje. V jiných prostředích se ignoruje.

Omezení pro metodu MQQueueManager.begin ()
Tuto metodu lze použít pouze pro správce front IBM MQ v systémech AIX, Linux, and Windows v režimu vazeb. Jinak se nezdaří s chybou MQRC_ENVIRONMENT_ERROR.

Další informace viz část "JTA/JDBC koordinace pomocí IBM MQ classes for Java" na stránce 387.

Variace v polích voleb MQGetMessage
Někteří správci front nepodporují strukturu MQGMO verze 2, takže musíte nastavit některá pole na jejich výchozí hodnoty.

Při použití správce front, který nepodporuje strukturu MQGMO verze 2, ponechte následující pole nastavena na výchozí hodnoty:

GroupStatus
SegmentStatus
Segmentace

Pole MatchOptions také podporuje pouze MQMO_MATCH_MSG_ID a MQMO_MATCH_CORREL_ID. Pokud do těchto polí vložíte nepodporované hodnoty, následné volání MQDestination.get() se nezdaří s chybou MQRC_GMO_ERROR. Pokud správce front nepodporuje strukturu MQGMO verze 2, tato pole se po úspěšné funkci MQDestination.get() neaktualizují.

Omezení v distribučních seznamech v adresáři IBM MQ classes for Java
Ne všichni správci front umožňují otevřít seznam MQDistributionList.

K vytvoření rozdělovníků se používají následující třídy:

MQDistributionList
Položka MQDistributionList
MQMessageTracker

Můžete vytvořit a naplnit položky MQDistributionLists a MQDistributionListv libovolném prostředí, ale ne všichni správci front vám umožňují otevřít MQDistributionList. Správci front z/OS zejména nepodporují distribuční seznamy. Při pokusu o otevření seznamu MQDistributionList při použití takového správce front dojde k chybě MQRC_OD_ERROR.

Variace v polích voleb MQPutMessage
Pokud správce front nepodporuje distribuční seznamy, s určitými poli MQPMO se zachází odlišně.

Čtyři pole v MQPMO jsou vykreslena jako následující členské proměnné ve třídě voleb MQPutMessage:

KnownDestCount
UnknownDestCount
InvalidDestCount
recordFields

Tato pole jsou primárně určena pro použití s rozdělovníky. Správce front, který podporuje distribuční seznamy, však také vyplní pole DestCount po MQPUT do jedné fronty. Pokud se například fronta interpretuje jako lokální fronta, hodnota knownDestCount je nastavena na 1 a další dvě pole počtu jsou nastavena na 0.

Pokud správce front nepodporuje distribuční seznamy, jsou tyto hodnoty simulovány následujícím způsobem:

- Pokud je funkce put () úspěšná, unknownDestPočet je nastaven na hodnotu 1 a ostatní jsou nastaveny na hodnotu 0.
- Pokud funkce put () selže, hodnota invalidDestCount je nastavena na 1 a ostatní jsou nastaveny na 0.

Proměnná recordFields se používá s rozdělovníky. Hodnotu lze kdykoli zapsat do pole recordFields bez ohledu na prostředí. Ignoruje se, pokud je objekt voleb MQPutMessagepoužit na následném objektu MQDestination.put() nebo MQQueueManager.put (), a nikoli MQDistributionList.put ().

Omezení v polích MQMD s IBM MQ classes for Java
Některá pole MQMD, která se týkají segmentace zpráv, by měla být ponechána na výchozí hodnotě při použití správce front, který segmentaci nepodporuje.

Následující pole MQMD se do značné míry zabývají segmentací zpráv:

GroupId
MsgSeqNumber
Offset
MsgFlags
OriginalLength

Pokud aplikace nastaví kterékoli z těchto polí MQMD na jiné hodnoty, než jsou jejich výchozí hodnoty, a poté provede operaci put () nebo get () ve správci front, který je nepodporuje, funkce put () nebo get () vyvolá výjimku MQException s hodnotou MQRC_MD_ERROR. Úspěšná metoda put () nebo get () s takovým správcem front vždy ponechá pole MQMD nastavená na výchozí hodnoty. Neodesílejte seskupenou nebo segmentovanou zprávu do aplikace Java , která je spuštěna pro správce front, který nepodporuje seskupení a segmentaci zpráv.

Pokud se aplikace Java pokusí získat () zprávu ze správce front, který tato pole nepodporuje, a fyzická zpráva, která má být načtena, je součástí skupiny segmentovaných zpráv (tj. má jiné než výchozí hodnoty pro pole MQMD), bude načtena bez chyb. Pole MQMD ve zprávě MQMessage však nejsou aktualizována,

vlastnost formátu MQMessage je nastavena na hodnotu MQFMT_MD_EXTENSION a skutečná data zprávy mají předponu se strukturou MQMDE, která obsahuje hodnoty pro nová pole.

Omezení pro IBM MQ classes for Java pod CICS Transakčním serverem

V prostředí CICS Transaction Server for z/OS může volání CICS nebo IBM MQ vydávat pouze hlavní (první) podproces.

Všimněte si, že třídy IBM MQ JMS nejsou podporovány pro použití v aplikaci CICS Java .

Proto není možné sdílet objekty MQQueueManager nebo MQQueue mezi podprocesy v tomto prostředí nebo vytvořit nový MQQueueManager v podřízeném podprocesu.

z/OS "Různé rozdíly mezi IBM MQ classes for Java na z/OS a jiných platformách" na stránce 421 identifikuje některá omezení a variace, které se vztahují na produkt IBM MQ classes for Java při spuštění pro správce front z/OS . Při spuštění v adresáři CICSnavíc nejsou podporovány metody řízení transakcí v produktu MQQueueManager . Namísto zadání MQQueueManager.commit () nebo MQQueueManager.backout () používají aplikace metody synchronizace úloh JCICS , Task.commit() a Task.rollback(). Třídu úloh dodává JCICS v balíku com.ibm.cics.server .

Použití adaptéru prostředků IBM MQ

Adaptér prostředků umožňuje aplikacím spuštěným na aplikačním serveru přistupovat k prostředkům IBM MQ . Podporuje příchozí a odchozí komunikaci.

Co adaptér prostředků obsahuje

V 9.3.0 V 9.3.0 V produktu IBM MQ 9.3.0je produkt Jakarta Messaging 3.0 podporován pro vývoj nových aplikací. Produkt IBM MQ 9.3.0 nadále podporuje produkt JMS 2.0 pro existující aplikace. Kromě adaptéru prostředků, který podporuje Java EE a JMS 2.0, IBM MQ 9.3.0 poskytuje adaptér prostředků, který podporuje Jakarta Messaging.

JM 3.0 **IBM MQ adaptér prostředků pro Jakarta Messaging**

Produkt Jakarta Connectors Architecture poskytuje standardní způsob připojení aplikací spuštěných v prostředí Jakarta EE k podnikovému informačnímu systému (EIS), například IBM MQ nebo Db2. Adaptér prostředků IBM MQ pro produkt Jakarta Messaging implementuje rozhraní Jakarta Connectors 2.0.0 a obsahuje rozhraní IBM MQ classes for Jakarta Messaging. Umožňuje aplikacím systému Jakarta Messaging a objektům typu message-driven bean (MDB) spuštěným na aplikačním serveru přistupovat k prostředkům správce front systému IBM MQ . Adaptér prostředků podporuje doménu typu point-to-point i doménu publikování/odběru.

JMS 2.0 **IBM MQ adaptér prostředků pro JMS 2.0**

Java Platform, Enterprise Edition Connector Architecture (JCA) poskytuje standardní způsob připojení aplikací spuštěných v prostředí Java EE k podnikovému informačnímu systému (EIS), například IBM MQ nebo Db2. Adaptér prostředků IBM MQ pro produkt JMS 2.0 implementuje rozhraní JCA 1.7 a obsahuje rozhraní IBM MQ classes for JMS. Umožňuje aplikacím systému JMS a objektům typu message-driven bean (MDB) spuštěným na aplikačním serveru přistupovat k prostředkům správce front systému 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 odešle JMS zprávy do cílů JMS a přijme JMS zprávy z cílů JMS synchronním způsobem.

Příchozí komunikace

Zpráva JMS , která dorazí do cíle systému JMS , je doručena do objektu MDB, který zpracovává zprávu asynchronně.

Adaptér prostředků také obsahuje soubor IBM MQ classes for Java. Třídy jsou automaticky k dispozici aplikacím, které jsou spuštěny na aplikačním serveru, do kterého byl adaptér prostředků implementován,

a umožňují aplikacím, které jsou spuštěny na tomto aplikačním serveru, používat rozhraní API IBM MQ classes for Java při přístupu k prostředkům správce front systému IBM MQ .

Použití IBM MQ classes for Java v rámci prostředí Java EE je podporováno s omezeními. Informace o těchto omezeních viz “Spuštění aplikací IBM MQ classes for Java v rámci Java EE” na stránce 337.

Která verze adaptéru prostředků se má použít

V 9.3.0 V 9.3.0 Verze adaptéru prostředků, kterou používáte, závisí na tom, zda jej implementujete na aplikační server, který podporuje Jakarta EE nebo Java EE:

V 9.3.0 JM 3.0 V 9.3.0 **Jakarta EE**

Od IBM MQ 9.3.0je podporován Jakarta Messaging 3.0 . Adaptér prostředků IBM MQ pro Jakarta Messaging musí být implementován v rámci aplikačního serveru, který podporuje Jakarta EE.

Java EE 7

Adaptér prostředků IBM MQ 8.0 a novější podporuje verzi JCA v1.7 a poskytuje podporu JMS 2.0 . Tento adaptér prostředků musí být implementován v rámci aplikačního serveru Java EE 7 a novějšího (viz “Příkaz podpory adaptéru prostředků IBM MQ” na stránce 425).

Adaptér prostředků IBM MQ 8.0 nebo novější můžete nainstalovat na libovolný aplikační server, který je certifikován jako vyhovující specifikaci Java Platform, Enterprise Edition 7 . Pomocí adaptéru prostředků IBM MQ 8.0 nebo novějšího se může aplikace připojit ke správci front pomocí přenosu BINDINGS nebo CLIENT.

Důležité: Adaptér prostředků IBM MQ 8.0 nebo novější lze implementovat pouze na aplikační server, který podporuje produkt JMS 2.0.

Použití adaptéru prostředků s produktem WebSphere Application Server traditional

V produktu IBM MQ 9.0je adaptér prostředků IBM MQ předinstalován v produktu WebSphere Application Server traditional 9.0 nebo novějším. Proto není nutné instalovat nový adaptér prostředků.

JM 3.0 WebSphere Application Server traditional v současné době nepodporuje Jakarta EE. Viz IBM MQ prohlášení o podpoře adaptéru prostředků.

Poznámka: Adaptér prostředků IBM MQ 9.0 nebo novější se může připojit v režimu transportu CLIENT nebo BINDINGS k libovolnému správci front IBM MQ ve službě.

Použití adaptéru prostředků s produktem WebSphere Liberty

Chcete-li se připojit k IBM MQ z WebSphere Liberty, musíte použít adaptér prostředků IBM MQ . Protože produkt Liberty neobsahuje adaptér prostředků IBM MQ , musíte jej získat odděleně od Fix Central.

V 9.3.0 V 9.3.0 Verze adaptéru prostředků, kterou používáte, závisí na tom, zda jej implementujete do verze produktu Liberty , která podporuje produkt Jakarta EE nebo Java EE.

Další informace o tom, jak stáhnout a nainstalovat adaptér prostředků, viz “Instalace adaptéru prostředků v adresáři Liberty” na stránce 432.

Související pojmy

“Konfigurace adaptéru prostředků pro příchozí komunikaci” na stránce 439
Chcete-li konfigurovat příchozí komunikaci, definujte vlastnosti jednoho nebo více objektů ActivationSpec .

“Konfigurace adaptéru prostředků pro odchozí komunikaci” na stránce 457
Chcete-li konfigurovat odchozí komunikaci, definujte vlastnosti objektu ConnectionFactory a spravovaného cílového objektu.

“Použití IBM MQ classes for JMS/Jakarta Messaging” na stránce 78
IBM MQ classes for JMS a IBM MQ classes for Jakarta Messaging jsou poskytovatelé systému zpráv Java dodávaní s produktem IBM MQ. Kromě implementace rozhraní definovaných ve specifikacích JMS

a Jakarta Messaging tito poskytovatelé systému zpráv přidávají do rozhraní API systému zpráv Java dvě sady rozšíření.

“Použití produktu IBM MQ classes for Java” na stránce 335
Použijte IBM MQ v prostředí Java . IBM MQ classes for Java Povolit aplikaci Java připojit se k produktu IBM MQ jako klient IBM MQ nebo se připojit přímo ke správci front IBM MQ .

Související odkazy

Konfigurace aplikačního serveru pro použití nejnovější úrovně údržby adaptéru prostředků

Určování problémů pro adaptér prostředků IBM MQ

WebSphere Application Server Témata

Údržba adaptéru prostředků IBM MQ

Implementace aplikací JMS na server Liberty pro použití poskytovatele systému zpráv IBM MQ

Příkaz podpory adaptéru prostředků IBM MQ

Produkt Adaptér prostředků IBM MQ , který musíte použít pro komunikaci mezi aplikací a správcem front, závisí na tom, zda používáte rozhraní API produktu Jakarta Messaging 3.0 nebo JMS 2.0 API.

JMS 2.0 Produkt IBM MQ 8.0 nebo novější se dodává s adaptérem prostředků, který implementuje specifikaci JMS 2.0 . Lze jej implementovat pouze na aplikační server, který je Java Platform, Enterprise Edition 7 (Java EE 7) vyhovující, a proto podporuje JMS 2.0. Seznam certifikovaných aplikačních serverů pro produkt Java Platform, Enterprise Edition je udržován na Oracle.

V 9.3.0 JM 3.0 V 9.3.0 V produktu IBM MQ 9.3.0je produkt Jakarta Messaging 3.0 podporován pro vývoj nových aplikací. Produkt IBM MQ 9.3.0 nadále podporuje produkt JMS 2.0 pro existující aplikace. Kromě adaptéru prostředků, který podporuje Java EE a JMS 2.0, IBM MQ 9.3.0 poskytuje adaptér prostředků, který podporuje Jakarta Messaging. Použití rozhraní API Jakarta Messaging 3.0 a rozhraní API JMS 2.0 ve stejné aplikaci není podporováno. Další informace viz Použití produktu IBM MQ Classes for JMS.

Nasazení v rámci WebSphere Liberty

WebSphere Liberty 8.5.5 Fix Pack 6 a novější a WebSphere Application Server Liberty 9.0 a novější jsou Java EE 7 certifikované aplikační servery, takže do nich lze implementovat adaptér prostředků IBM MQ 9.0 .

V 9.3.0 V 9.3.0 Chcete-li použít adaptér prostředků IBM MQ pro produkt Jakarta Messaging s produktem Liberty, musíte použít verzi produktu Liberty , která podporuje produkt Jakarta EE.

Produkt WebSphere Liberty má k dispozici následující funkce pro práci s adaptéry prostředků:

- V 9.3.0 JM 3.0 V 9.3.0 Funkce messaging-3.0 umožňuje pracovat s adaptéry prostředků Jakarta Messaging 3.0 .
- Funkce wmqJmsClient-2.0 umožňuje práci s adaptéry prostředků JMS 2.0 .
- Funkce wmqJmsClient-1.1 umožňuje práci s adaptéry prostředků JMS 1.1 .

Důležité:

- V 9.3.0 JM 3.0 V 9.3.0 Adaptér prostředků IBM MQ pro produkt Jakarta Messaging musí být implementován do verze produktu Liberty , která podporuje produkt Jakarta EE. Tento adaptér prostředků nelze použít s verzemi produktu Liberty , které podporují starší Java EE specifikaci, nikoli Jakarta EE.
- JMS 2.0 Adaptér prostředků IBM MQ 8.0 nebo novější, který podporuje JMS 2.0 , musí být implementován s funkcí wmqJmsClient-2.0 .

Nasazení v rámci WebSphere Application Server traditional

Produkt WebSphere Application Server traditional 9.0 je dodáván s již nainstalovaným adaptérem prostředků IBM MQ 9.0 . Proto není nutné instalovat nový adaptér prostředků. Instalovaný adaptér prostředků se může připojit v režimu transportu CLIENT nebo BINDINGS k libovolným správcům front, kteří jsou spuštěni v podporované verzi produktu IBM MQ. Další informace viz “Konektivita ke správcům front IBM MQ 8.0 nebo novějším” na stránce 426.

Důležité:

- Adaptér prostředků IBM MQ 9.0 nelze implementovat do verzí produktu WebSphere Application Server traditional před IBM MQ 9.0, protože tyto verze nejsou Java EE 7 certifikovány.
- JM 3.0 WebSphere Application Server traditional v současné době nepodporuje Jakarta EE.

Další informace o verzích adaptéru prostředků dodávaných s produktem WebSphere Application Servernaleznete v technické poznámce Která verze adaptéru prostředků produktu WebSphere MQ je dodávána s produktem WebSphere Application Server?

Použití adaptéru prostředků s jinými aplikačními servery

U všech ostatních aplikačních serverů kompatibilních s produktem Java EE 7 nebo Jakarta EE lze problémy, které se vyskytnou po úspěšném dokončení IBM MQ adaptéru prostředků Test ověření instalace (IVT), nahlásit do produktu IBM pro vyšetřování trasování produktu IBM MQ a dalších IBM MQ diagnostických informací. Pokud nelze IVT adaptéru prostředků IBM MQ úspěšně spustit, všechny zjištěné problémy mohou být způsobeny nesprávnou implementací nebo chybnými definicemi prostředků, které jsou specifické pro aplikační server, a problémy musí být prozkoumány pomocí dokumentace aplikačního serveru a organizace podpory pro tento aplikační server.

Java Běhové prostředí

Prostředí JRE (Java Runtime), které se používá ke spuštění aplikačního serveru, musí být takové, které je podporováno klientem IBM MQ 9.0 nebo novějším. Další informace viz téma Systémové požadavky pro IBM MQ. (Vyberte, kterou verzi a sestavu operačního systému nebo komponenty chcete zobrazit, pak postupujte podle odkazu **Java** , který je uveden na kartě **Podporovaný software** .)

Konektivita ke správcům front IBM MQ 8.0 nebo novějším

Úplný rozsah funkcí produktu JMS 2.0 je k dispozici při připojování ke správci front produktu IBM MQ 8.0 nebo novějšímu pomocí adaptéru prostředků, který byl implementován na certifikovaný aplikační server Java EE 7 . Chcete-li využít funkce JMS 2.0 , musí se adaptér prostředků připojit ke správci front pomocí normálního režimu poskytovatele systému zpráv IBM MQ . Další informace viz Konfigurace vlastnosti JMS **PROVIDERVERSION**.

V 9.3.0 JM 3.0 V 9.3.0 Úplný rozsah funkcí produktu Jakarta Messaging 3.0 je k dispozici při připojování ke správci front produktu IBM MQ 9.3 pomocí adaptéru prostředků, který byl implementován na certifikovaný aplikační server Jakarta EE .

Rozšíření produktu MQ

Specifikace JMS 2.0 zavádí změny v tom, jak určité chování funguje. Vzhledem k tomu, že produkt IBM MQ 8.0 nebo novější implementuje tuto specifikaci, dochází ke změnám v chování mezi produktem IBM MQ 8.0 a novějšími a dřívějšími verzemi produktu. V produktu IBM MQ 8.0 nebo novějším produkt IBM MQ classes for JMS zahrnuje podporu pro Java systémovou vlastnost `com.ibm.mq.jms.SupportMQExtensions` , která při nastavení na hodnotu TRUEzpůsobí, že tyto verze produktu IBM MQ vrátí tato chování zpět na chování produktu IBM WebSphere MQ 7.5 nebo starší. Výchozí hodnota vlastnosti je FALSE.

Adaptér prostředků IBM MQ 9.0 nebo novější také obsahuje vlastnost adaptéru prostředků s názvem `supportMQExtensions` , která má stejný efekt a výchozí hodnotu jako systémová vlastnost

`com.ibm.mq.jms.SupportMQExtensions` Java . Tato vlastnost adaptéru prostředků je ve výchozím nastavení v souboru `ra.xml` nastavena na hodnotu false.

Pokud je nastavena vlastnost adaptéru prostředků i systémová vlastnost Java , má přednost systémová vlastnost.

Všimněte si, že v rámci adaptéru prostředků, který je již implementován v rámci produktu WebSphere Application Server traditional 9.0, je tato vlastnost automaticky nastavena na hodnotu TRUE , což pomáhá při migraci.

Další informace viz téma “Vlastnost SupportMQExtensions” na stránce 317.

Obecné otázky

Prokládané relace nejsou podporovány
Některé aplikační servery poskytují schopnost nazývanou prokládáním relací, kde lze stejnou relaci JMS použít ve více transakcích, i když je uvedena pouze v jednom okamžiku. Adaptér prostředků IBM MQ nepodporuje tuto schopnost, což může vést k následujícím problémům:

```
Pokus o vložení zprávy do fronty MQ se nezdařil s kódem příčiny 2072
(MQRC_SYNCPOINT_NOT_AVAILABLE).
```

Volání funkce xa_close () selže s kódem příčiny -3 (XAER_PROTO) a na správci front IBM MQ , ke kterému se přistupuje z aplikačního serveru, se vygeneruje FDC s ID zkoušky AT040010 . Informace o tom, jak tuto funkci zakázat, naleznete v dokumentaci k aplikačnímu serveru.

Specifikace Java Transaction API (JTA) způsobu zotavení prostředků XA pro zotavení transakcí XA.
Část 3.4.8 specifikace JTA nedefinuje specifický mechanizmus, pomocí kterého jsou znovu vytvořeny prostředky XA k provedení transakčního zotavení XA. Jako takový je na každém jednotlivém správci transakcí (a tedy na aplikačním serveru), jak jsou prostředky XA zahrnuté do transakce XA obnoveny. U některých aplikačních serverů je možné, že adaptér prostředků IBM MQ 9.0 neimplementuje mechanizmy specifické pro aplikační server, které se používají k provádění zotavení transakcí XA.

Odpovídající připojení v továrně ManagedConnection
Aplikační server může vyvolat metodu matchManagedConnections v instanci továrny ManagedConnectionposkytované adaptérem prostředků IBM MQ . Funkce ManagedConnection je vrácena pouze v případě, že metoda nalezne argumenty **`javax.security.auth.Subject`** a **`javax.resource.spi.ConnectionRequestInfo`** , které byly aplikačním serverem předány metodě.

Omezení adaptéru prostředků IBM MQ

Adaptér prostředků IBM MQ je podporován na všech platformách IBM MQ . Pokud však používáte adaptér prostředků IBM MQ , některé funkce produktu IBM MQ nejsou k dispozici nebo jsou omezené.

Adaptér prostředků IBM MQ má následující omezení:

- Od IBM MQ 8.0je adaptér prostředku Java Platform, Enterprise Edition 7 (Java EE 7) adaptér prostředku poskytující funkci JMS 2.0 . V důsledku toho musí být adaptér prostředků IBM MQ 8.0 nebo novější nainstalován na certifikovaném aplikačním serveru Java EE 7 nebo novějším. Může se připojit v režimu transportu klienta nebo vazeb k libovolnému správci front v rámci služby.
- Při spuštění v rámci aplikačního serveru WebSphere Liberty nejsou stabilizované IBM MQ classes for Java podporovány. V rámci jiných aplikačních serverů se nedoporučuje používat produkt IBM MQ classes for Java . Podrobnosti o IBM MQ classes for Java aspektech Java EEviz IBM technická poznámka Použití rozhraní WebSphere MQ Java v prostředí J2EE/JEE .
- Při spuštění uvnitř aplikačního serveru WebSphere Liberty na systému z/OSmusí být použita funkce wmqJmsClient-2.0 . Generická podpora JCA není pro z/OSmožná.
- Adaptér prostředků IBM MQ nepodporuje programy uživatelské procedury kanálu, které jsou napsány v jiných jazycích než Java.
- Zatímco je aplikační server spuštěn, hodnota vlastnosti sslFipsRequired musí být true pro všechny prostředky JCA nebo false pro všechny prostředky JCA . Toto je požadavek, i když se prostředky JCA nepoužívají souběžně. Pokud má vlastnost sslFipsRequired různé hodnoty pro různé prostředky JCA ,

IBM MQ vydá kód příčiny MQRC_UNSUPPORTED_CIPHER_SUITE, a to i v případě, že není použito připojení TLS.

- Pro aplikační server nelze zadat více než jedno úložiště klíčů. Jsou-li vytvořena připojení k více než jednomu správci front, musí všechna připojení používat stejné úložiště klíčů. Toto omezení se nevztahuje na WebSphere Application Server.
- Pokud použijete tabulku CCDT (Client Channel Definition Table) s více než jednou vhodnou definicí kanálu připojení klienta, v případě selhání může adaptér prostředků vybrat jinou definici kanálu, a tedy jiného správce front z tabulky CCDT, což by způsobilo problémy při zotavení transakce. Adaptér prostředků neprovádí žádnou akci, aby zabránil použití takové konfigurace, a je vaší odpovědností vyhnout se konfiguracím, které by mohly způsobit problémy s obnovou transakcí.
- Funkce opakování připojení není podporována pro odchozí připojení při spuštění v kontejneru Java EE (EJB/Servlet). Opakovaný pokus o připojení není vůbec podporován pro odchozí JMS , když je adaptér použit v kontextu kontejneru JEE , bez ohledu na konfiguraci transakce nebo pro netransakční použití.
- Opětovné ověření, jak je definováno v části 9.1.9 specifikace Java EE Connector Architecture verze 1.7 , připojení JMS není podporováno. Soubor `ra.xml` v rámci modulu pro adpater prostředků 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 IBM MQ vygeneruje výjimku javax.resource.spi.SecurityException s kódem zprávy MQJCA1028 .

Související úlohy

Určení, že za běhu jsou v klientu MQI použity pouze specifikace CipherSpecs s certifikací FIPS.

Související odkazy

Standard FIPS (Federal Information Processing Standards) pro AIX, Linux, and Windows

WebSphere Application Server a adaptér prostředků IBM MQ

Adaptér prostředků IBM MQ je používán aplikacemi, které provádějí systém zpráv JMS s poskytovatelem systému zpráv IBM MQ v produktu WebSphere Application Server.

Důležité: Nepoužívejte adaptér prostředků IBM MQ s WebSphere Application Server 6.0 nebo WebSphere Application Server 6.1.

Produkt WebSphere Application Server traditional 9.0 obsahuje verzi adaptéru prostředků IBM MQ 9.0 . Adaptér prostředků IBM MQ 9.0 nebo novější nelze implementovat do dřívějších verzí produktu WebSphere Application Server, protože tyto verze nejsou Java EE 7 certifikovány.

WebSphere Application Server traditional v současné době nepodporuje Jakarta EE. Viz IBM MQ prohlášení o podpoře adaptéru prostředků.

Chcete-li použít aplikaci JMS pro přístup k prostředkům správce front IBM MQ z produktu WebSphere Application Server, použijte poskytovatele systému zpráv IBM MQ v adresáři WebSphere Application Server. Poskytovatel systému zpráv IBM MQ obsahuje verzi produktu IBM MQ classes for JMS. Další informace naleznete v technické poznámce Která verze produktu WebSphere MQ Resource Adapter (RA) je dodávána s produktem WebSphere Application Server?.

Důležité: Nezahrnujte žádný ze souborů JAR IBM MQ classes for JMS nebo IBM MQ classes for Java do své aplikace. To může vést k výjimkám ClassCasta může být obtížné je udržovat.

Liberty a adaptér prostředků IBM MQ

Adaptér prostředků IBM MQ lze nainstalovat do produktu WebSphere Liberty pomocí funkce Liberty . Funkce, kterou používáte, závisí na verzi adaptéru prostředků, kterou instalujete. Alternativně můžete, s výhradou některých omezení, nainstalovat adaptér prostředků pomocí generické podpory Java Platform, Enterprise Edition Connector Architecture (Java EE JCA).

Obecná omezení při instalaci adaptéru prostředků do adresáře Liberty

Následující omezení platí pro adaptér prostředků při použití funkce wmqJmsClient-1.1 nebo wmqJmsClient-2.0 a také při použití generické podpory JCA :

- IBM MQ classes for Java nejsou v produktu Libertypodporovány. Nesmí se používat s funkcí systému zpráv IBM MQ Liberty ani s generickou podporou JCA . Další informace naleznete v tématu Použití rozhraní WebSphere MQ Java v prostředí J2EE/JEE.
- Adaptér prostředků IBM MQ má typ transportu BINDINGS_THEN_CLIENT. Tento typ transportu není podporován v rámci funkce systému zpráv produktu IBM MQ Liberty .
- Před produktem IBM MQ 9.0nebyla funkce Advanced Message Security (AMS) součástí funkce systému zpráv produktu IBM MQ Liberty . Produkt AMS je však podporován s adaptérem prostředků IBM MQ 9.0 nebo novějším.

Poznámka: V systémech IBM MQ verze vyšší než IBM MQ 9.0.0.6 a IBM MQ 9.1.0.1 byste měli místo funkce ssl-1.0 použít funkci transportSecurity-1.0 .

Další informace naleznete v následujících tématech:

Povolení komunikace SSL v produktu Liberty
Výchozí nastavení SSL v Liberty
Zabezpečení přenosu 1.0

Omezení při použití funkcí Liberty

S volbou WebSphere Liberty 8.5.5 Fix Pack 2 až WebSphere Liberty 8.5.5 Fix Pack 5 včetně byla k dispozici pouze funkce wmqJmsClient-1.1 a bylo možné použít pouze funkci JMS 1.1 . WebSphere Liberty 8.5.5 Fix Pack 6 přidal funkci wmqJmsClient-2.0 , aby bylo možné použít JMS 2.0 .

V 9.3.0 JM 3.0 V 9.3.0 V produktu IBM MQ 9.3.0je podporována hodnota Jakarta Messaging 3.0 . Chcete-li použít adaptér prostředků IBM MQ pro produkt Jakarta Messaging s produktem Liberty, musíte použít verzi produktu Liberty , která podporuje produkt Jakarta EE. Musíte použít adaptér prostředků pro systém Jakarta Messaging s funkcí Liberty generic messaging-3.0 .

Funkce, kterou musíte použít, závisí na verzi adaptéru prostředků, kterou používáte:

- Adaptér prostředků IBM MQ 8.0.0 Fix Pack 3 a novější IBM MQ 8.0 lze použít pouze s funkcí wmqJmsClient-2.0 .
- Adaptér prostředků IBM MQ 9.0 lze použít pouze s funkcí wmqJmsClient-2.0 .
- V 9.3.0 JM 3.0 V 9.3.0 Funkce messaging-3.0 umožňuje pracovat s adaptéry prostředků Jakarta Messaging 3.0 .

Omezení při použití generické podpory JCA

Pokud používáte generickou podporu JCA , platí následující omezení:

- Musíte uvést úroveň JMS , když používáte generickou podporu JCA . JMS 2.0 a JCA 1.7 musí být použity pouze s adaptéry prostředků IBM MQ 8.0.0 Fix Pack 3 a novější IBM MQ 8.0 .
- Není možné spustit adaptér prostředků IBM MQ na systému z/OS s použitím generické podpory JCA . Chcete-li spustit adaptér prostředků 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:

JM 3.0
```
<resourceAdapter id="mqJms" location="${server.config.dir}/
wmq.jakarta.jmsra.rar">
  <classloader apiTypeVisibility="spec, ibm-api, api, third-party"/>
</resourceAdapter>
```

JMS 2.0
```
<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 KROMĚ hodnoty `wmqJms`. Pokud použijete `wmqJms` jako ID, pak produkt Liberty nebude schopen řádně načíst adaptér prostředků. Důvodem je, že `wmqJms`

je ID, které se interně používá k odkazování na specifickou funkci pro IBM MQ. Ve skutečnosti vytváří výjimku NullPointer.

Následující příklady ukazují některé úseky kódu ze souboru `server.xml` při spuštění příkazu JMS 2.0:

```
<!-- 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 nedostatku 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 upřednostňováno. Nepoužívejte soubor `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, že classloaderProviderRef zpět na adaptér prostředků prostřednictvím ID `mqJms`; to znamená povolit načtení tříd specifických pro IBM MQ.

Omezení při trasování pomocí generické podpory JCA

Trasování a protokolování nejsou integrovány do systému trasování Liberty . Místo toho musí být trasování adaptéru prostředků IBM MQ povoleno buď pomocí systémových vlastností Java , nebo pomocí konfiguračního souboru IBM MQ classes for JMS , jak je popsáno v tématu Trasování IBM MQ classes for JMS aplikací. Podrobnosti o nastavení vlastností systému Java v souboru Libertynaleznete v WebSphere Liberty dokumentaci.

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 do tohoto souboru následující volby prostředí JVM, abyste povolili trasování na jeden řádek:

   ```
   -Dcom.ibm.msg.client.commonservices.trace.status=ON
   -Dcom.ibm.msg.client.commonservices.trace.outputName=C:\Trace\MQRA-WLP_%PID%.trc
   ```

3. Chcete-li použít tato nastavení na jeden server, uložte soubor `jvm.options` na adrese:

   ```
   ${server.config.dir}/jvm.options
   ```

   Chcete-li použít tyto změny na všechny Liberty, uložte `jvm.options` na:

   ```
   ${wlp.install.dir}/etc/jvm.options
   ```

   Toto se projeví pro všechna prostředí JVM, která nemají lokálně definovaný soubor `jvm.options` .
4. Restartujte server, abyste povolili změny.

To má za následek zápis trasování do trasovacího souboru s názvem `MQRA-WLP_<process identifier>.trc` v adresáři `<path_to_trace_to>`.

Úplná podpora produktu Liberty XA s definičními tabulkami kanálů klienta

Používáte-li produkt WebSphere Liberty 18.0.0.2 dále s produktem IBM MQ 9.2.0 nebo novějším, můžete ve spojení s transakcemi XA používat skupiny správců front v tabulce CCDT (Client Channel Definition

Table). To znamená, že je nyní možné využívat distribuci a dostupnost pracovní zátěže poskytované skupinami správců front při zachování integrity transakcí.

V případě chyb připojení ke správci front musí být správce front znovu k dispozici, aby bylo možné transakci vyřešit. Zotavení transakce je spravováno produktem Libertya možná budete muset nakonfigurovat správce transakcí tak, aby byla pro správce front znovu k dispozici odpovídající doba. Další informace viz Správce transakcí (transakce) v dokumentaci k produktu WebSphere Liberty .

Jedná se o funkci na straně klienta, tj. potřebujete adaptér prostředků IBM MQ 9.2.0 nebo novější, nikoli správce front IBM MQ 9.2.0 nebo novější.

Instalace adaptéru prostředků IBM MQ

Adaptér prostředků IBM MQ je dodáván jako soubor archivu prostředků (RAR). Nainstalujte soubor RAR na aplikační server. Možná budete muset přidat adresáře do systémové cesty.

Informace o této úloze

Adaptér prostředků IBM MQ je dodáván jako soubor archivu prostředků (RAR):

- V 9.3.0 JM 3.0 V 9.3.0 Pro systém Jakarta Messaging 3.0se tento soubor nazývá `wmq.jakarta.jmsra.rar`. Soubor RAR obsahuje IBM MQ classes for Jakarta Messaging a IBM MQ implementaci rozhraní Jakarta Connectors Architecture (JCA).
- JMS 2.0 Pro systém JMS 2.0se tento soubor nazývá `wmq.jmsra.rar`. Soubor RAR obsahuje IBM MQ classes for JMS a implementaci IBM MQ rozhraní Java EE Connector Architecture (JCA).

Když instalujete adaptér prostředků jako součást instalace produktu IBM MQ , soubor RAR se nainstaluje s produktem IBM MQ classes for JMS v adresáři uvedeném v části Tabulka 61 na stránce 431.

Tabulka 61. Adresář IBM MQ obsahující soubor RAR pro každou platformu

Platforma	**Adresář**
AIX and Linux	*MQ_INSTALLATION_PATH*/java/lib/jca
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 .

Musíte použít adaptér prostředků IBM MQ pro připojení k produktu IBM MQ z aplikačního serveru. V závislosti na tom, který aplikační server používáte, může být adaptér prostředků předinstalován nebo jej budete muset nainstalovat sami.

Tabulka 62. Instalace adaptéru prostředků na aplikačním serveru

Aplikační server	**Máte předinstalovanou instalaci nebo ji potřebujete nainstalovat?**
WebSphere Application Server traditional 9.0	Adaptér prostředků IBM MQ 9.0 je předinstalován v produktu WebSphere Application Server traditional 9.0. Proto nemusíte instalovat nový adaptér prostředků do adresáře WebSphere Application Server traditional 9.0.
WebSphere Liberty	Produkt WebSphere Liberty neobsahuje adaptér prostředků IBM MQ , takže jej musíte získat odděleně od Fix Central.

Tabulka 62. Instalace adaptéru prostředků na aplikačním serveru (pokračování)	
Aplikační server	**Máte předinstalovanou instalaci nebo ji potřebujete nainstalovat?**
Jiný aplikační server Java EE nebo Jakarta EE	Získejte adaptér prostředků odděleně od produktu Fix Central, jak je tomu v případě produktu WebSphere Liberty.

Procedura

- Pokud se připojujete k produktu IBM MQ z produktu WebSphere Libertynebo jiného aplikačního serveru Java EE nebo Jakarta EE , stáhněte a nainstalujte adaptér prostředků IBM MQ , jak je popsáno v tématu “Instalace adaptéru prostředků v adresáři Liberty” na stránce 432.
- Linux AIX
  U připojení vazeb v systémech AIX and Linux se ujistěte, že adresář obsahující knihovny JNI (Java Native Interface) je v systémové cestě.
  Umístění tohoto adresáře, který také obsahuje knihovny IBM MQ classes for JMS , viz “Konfigurace knihoven JNI (Java Native Interface)” na stránce 92.
  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 souboru WebSphere Liberty by to bylo nakonfigurováno tak, jak je uvedeno v následujícím příkladu:

```
<wmqJmsClient nativeLibraryPath="/opt/mqm/java/lib64"/>
```

  Transakce jsou podporovány v režimu klienta i v režimu vazeb.

Instalace adaptéru prostředků v adresáři Liberty

Chcete-li se připojit k produktu IBM MQ z produktu WebSphere Libertynebo z jiných aplikačních serverů Java EE nebo Jakarta EE , musíte použít adaptér prostředků IBM MQ . Protože produkt Liberty neobsahuje adaptér prostředků IBM MQ , musíte jej získat odděleně od Fix Central.

Než začnete

Poznámka: Informace v tomto tématu se nevztahují na WebSphere Application Server traditional 9.0. Adaptér prostředků IBM MQ 9.0 je předinstalován v 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 máte v počítači nainstalováno prostředí Java runtime environment (JRE) a že bylo prostředí JRE přidáno do systémové cesty.

Instalační program produktu Java , který se používá v tomto procesu instalace, nevyžaduje spuštění jako uživatel root ani žádný konkrétní uživatel. Jediným požadavkem je, aby uživatel, který jej spouští, měl přístup pro zápis do adresáře, do kterého chcete soubory vjet.

V případě verzí systému Liberty až do verze WebSphere Liberty 8.5.5 Fix Pack 1platí, že pokud je objekt EJB implementován pouze s použitím konfigurace v rámci produktu `ejb-jar.xml`, musí být pro verzi produktu WebSphere Application Server , kterou používá profil produktu Liberty , použita oprava APAR PM89890 . Tato metoda konfigurace se používá pro program pro ověření instalace (IVT) adaptéru prostředků, takže tato oprava APAR je nezbytná pro spuštění IVT.

V 9.3.0 JM 3.0 V 9.3.0 V produktu IBM MQ 9.3.0je podporována hodnota Jakarta Messaging 3.0 . Chcete-li použít adaptér prostředků IBM MQ pro produkt Jakarta Messaging s produktem Liberty, musíte použít verzi produktu Liberty , která podporuje produkt Jakarta EE. Můžete například použít funkci Liberty generic messaging-3.0 .

Informace o této úloze

Soubor JAR pro adaptér prostředků, který můžete stáhnout z adresáře Fix Central , je spustitelný. Když spustíte tento spustitelný soubor, zobrazí se licenční smlouva IBM MQ , která musí být přijata. Požádá o adresář, do kterého se má nainstalovat adaptér prostředků IBM MQ . Do tohoto adresáře se pak nainstaluje soubor RAR adaptéru prostředků a program IVT (installation verification test). Můžete buď přijmout předvolbu, nebo zadat jiný adresář, kterým může být adresář adaptérů prostředků aplikačního serveru, nebo jakýkoli jiný adresář ve vašem systému. Adresář se vytvoří jako součást instalace, pokud neexistuje.

Před IBM MQ 9.0byl název souboru, který se má stáhnout, 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 jej můžete konfigurovat v adresáři WebSphere Liberty.

Postup

1. Stáhněte adaptér prostředků IBM MQ z adresáře Fix Central.
   a) Klepněte na tento odkaz: IBM MQ Adaptér prostředků.
   b) Vyhledejte adaptér prostředků pro svou verzi produktu IBM MQ v zobrazeném seznamu dostupných oprav.
      Příklad:

      ```
      release level: 9.1.4.0-IBM-MQ-Java-InstallRA
      Continuous Delivery Release: 9.1.4 IBM 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 stáhli soubor.
   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, úpravy a opravné sady a *V.R.M.F*-IBM-MQ-Java-InstallRA.jar je název souboru, který byl stažen z produktu Fix Central.
   Chcete-li například nainstalovat adaptér prostředků IBM MQ pro vydání IBM MQ 9.1.4 , použijte následující příkaz:

   ```
   java -jar 9.1.4.0-IBM-MQ-Java-InstallRA.jar
   ```

   Poznámka: Chcete-li provést tuto instalaci, musíte mít na svém počítači nainstalováno prostředí JRE a přidáno do systémové cesty.

   Když zadáte příkaz, zobrazí se následující informace:

   ```
   Než budete moci použít, extrahovat nebo nainstalovat produkt IBM MQ 9.1, musíte přijmout
   podmínky 1. IBM Mezinárodní licenční smlouva pro hodnocení
   Programy 2. IBM Mezinárodní licenční smlouva pro programy a další
   informace o licenci. Přečtěte si pozorně následující licenční smlouvy.

   Licenční smlouvu lze samostatně zobrazit pomocí
   --viewLicenseVolba smlouvy.
   Stiskněte klávesu Enter, chcete-li nyní zobrazit licenční podmínky, nebo 'x', chcete-li
   přeskočit.
   ```
3. Přezkoumejte a přijměte licenční podmínky:
   a) Chcete-li zobrazit licenci, stiskněte klávesu Enter.
      Případně stisknutím tlačítka x přeskočí zobrazení licence.

Po zobrazení licence nebo bezprostředně po výběru x se zobrazí následující zpráva, která vám sdělí, že si můžete zvolit zobrazení dalších licenčních podmínek:

```
Další informace o licenci lze samostatně zobrazit pomocí
--viewLicenseVolba informací.
Stiskněte klávesu Enter, chcete-li nyní zobrazit další informace o licenci, nebo 'x',
chcete-li přeskočit.
```

b) Chcete-li zobrazit další licenční podmínky, stiskněte klávesu Enter.

Případně stisknutí tlačítka x přeskočí zobrazení dalších licenčních podmínek.

Po zobrazení dalších licenčních podmínek nebo ihned po výběru x se zobrazí následující zpráva, která vás požádá o přijetí licenční smlouvy:

```
Výběrem možnosti "Souhlasím" níže souhlasíte s podmínkami
licenční smlouva a jiné podmínky nežIBM , jsou-li použitelné. Pokud ne,
souhlasím, vyberte "nesouhlasím".

Vyberte [ 1] Souhlasím, nebo [ 2] NeSouhlasím:
```

c) Chcete-li přijmout licenční smlouvu a pokračovat ve výběru instalačního adresáře, vyberte 1.

Vyberete-li volbu 2, instalace se ukončí okamžitě.

Pokud jste vybrali 1, zobrazí se následující zpráva, která vás požádá o výběr cílového instalačního adresáře:

```
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. Zadejte instalační adresář pro adaptér prostředků:

- Chcete-li instalovat adaptér prostředků do výchozího umístění, stiskněte klávesu Enter bez zadání hodnoty.
- Chcete-li instalovat adaptér prostředků do jiného umístění, než je výchozí, zadejte název adresáře, do kterého chcete instalovat adaptér prostředků, a stiskněte klávesu Enter.

Po instalaci souborů do vybraného umístění se zobrazí potvrzovací zpráva, jak ukazuje následující příklad:

```
Extrahování souborů do H: \Liberty\WMQ \wmq
Všechny soubory produktu se úspěšně extrahovaly.
```

Během instalace se ve vybraném instalačním adresáři vytvoří nový adresář s názvem `wmq` a následující soubory se pak nainstalují do adresáře `wmq` :

- Testovací program ověření instalace, `wmq.jakarta.jmsra.ivt` (Jakarta Messaging 3.0) nebo `wmq.jmsra.ivt` (JMS 2.0).
- Soubor IBM MQ RAR,`wmq.jakarta.jmsra.rar` (Jakarta Messaging 3.0 nebo `wmq.jmsra.rar` (JMS 2.0)).

5. JMS 2.0

Volitelné: Nakonfigurujte adaptér prostředků Java EE 7 (JMS 2.0) v adresáři WebSphere Liberty Profile.

Kroky, které musíte provést při konfiguraci adaptéru prostředků v souboru Liberty , jsou následující. Další informace viz WebSphere Application Server dokumentace k produktu.

a) Přidejte funkci wmqJmsClient-2.0 do souboru `server.xml` , abyste umožnili práci s adaptérem prostředků IBM MQ .

Další informace viz téma “Která verze adaptéru prostředků se má použít” na stránce 424.

b) Přidejte odkaz na soubor `wmq.jmsra.rar` (JMS 2.0), který jste nainstalovali.

Příklad konfigurace pro podporu servletů a objektů MDB s databází JNDI může vypadat takto:

```
<featureManager>
   <feature>wmqJmsClient-2.0</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"/>
```

6. JM 3.0

   Volitelné: Nakonfigurujte adaptér prostředků Jakarta EE 9 (Jakarta Messaging 3.0) v adresáři WebSphere Liberty Profile.

   Kroky, které musíte provést při konfiguraci adaptéru prostředků v souboru Liberty , jsou následující. Další informace viz WebSphere Application Server dokumentace k produktu.

   a) Přidejte funkci wmqJmsClient-3.0 do souboru `server.xml` , abyste umožnili práci s adaptérem prostředků IBM MQ .

      Další informace viz téma “Která verze adaptéru prostředků se má použít” na stránce 424.

   b) Přidejte odkaz na soubor `wmq.jakarta.jmsra.rar` (Jakarta Messaging 3.0), který jste nainstalovali.

   Příklad konfigurace pro podporu servletů a objektů MDB s databází JNDI může vypadat takto:

```
 <featureManager>
    <feature>wmqJmsClient-3.0</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"/>
```

   Poznámka: Používáte-li volbu Open Libertynamísto WebSphere Liberty Profile, budete muset použít funkci podpory generického adaptéru prostředků "messagingClient-3.0" namísto "wmqJmsClient-3.0" a ostatní aspekty konfigurace se budou lišit. Další podrobnosti naleznete v dokumentaci Open Liberty .

Konfigurace adaptéru prostředků IBM MQ

Chcete-li konfigurovat adaptér prostředků IBM MQ , definujte různé prostředky Java Platform, Enterprise Edition Connector Architecture (JCA) a volitelně systémové vlastnosti. Musíte také nakonfigurovat adaptér prostředků, aby spustil program IVT (installation verification test). To je důležité, protože služba IBM může vyžadovat spuštění tohoto programu, aby označila, že jakýkoli jiný aplikační server nežIBM byl správně nakonfigurován.

Než začnete

Tato úloha předpokládá, že jste již obeznámeni s JMS a IBM MQ classes for JMS. Mnoho vlastností použitých ke konfiguraci adaptéru prostředků IBM MQ je ekvivalentem vlastností objektů IBM MQ classes for JMS a má stejnou funkci.

Informace o této úloze

Každý aplikační server poskytuje vlastní sadu rozhraní administrace. Některé aplikační servery poskytují grafická uživatelská rozhraní pro definování prostředků JCA , ale jiné vyžadují, aby administrátor napsal plány implementace XML. Proto je mimo rozsah této dokumentace poskytnout 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 konfiguraci adaptéru prostředků JCA naleznete v dokumentaci dodané s aplikačním serverem.

Procedura

Definujte prostředky JCA v následujících kategoriích:

- Definujte vlastnosti objektu ResourceAdapter .

Tyto vlastnosti, které představují globální vlastnosti adaptéru prostředků, například úroveň diagnostického trasování, jsou popsány v tématu “Konfigurace pro vlastnosti objektu ResourceAdapter” na stránce 437.

- Definujte vlastnosti objektu ActivationSpec .

  Tyto vlastnosti určují, jak je objekt MDB aktivován pro příchozí komunikaci. Další informace viz téma “Konfigurace adaptéru prostředků pro příchozí komunikaci” na stránce 439.

- Definujte vlastnosti objektu ConnectionFactory .

  Aplikační server používá tyto vlastnosti k vytvoření objektu JMS ConnectionFactory pro odchozí komunikaci. Další informace viz téma “Konfigurace adaptéru prostředků pro odchozí komunikaci” na stránce 457.

- Definujte vlastnosti spravovaného cílového objektu.

  Aplikační server používá tyto vlastnosti k vytvoření JMS objektu fronty nebo JMS objektu tématu pro odchozí komunikaci. Další informace viz téma “Konfigurace adaptéru prostředků pro odchozí komunikaci” na stránce 457.

- Volitelné: Definujte plán implementace pro adaptér prostředků.

  Soubor RAR adaptéru prostředků IBM MQ obsahuje soubor s názvem `META-INF/ra.xml`, který obsahuje deskriptor implementace pro adaptér prostředků. Tento deskriptor implementace je definován schématem XML na adrese 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 nasazení je specifický pro aplikační server.

Určete systémové vlastnosti prostředí JVM podle potřeby:

- Používáte-li protokol TLS (Transport Layer Security), zadejte 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 nelze určit více než jedno úložiště klíčů pro aplikační server. Vlastnosti se vztahují na celé prostředí JVM, a proto mohou ovlivnit aplikační server, pokud jiné aplikace spuštěné na aplikačním serveru používají připojení TLS. Aplikační server může také resetovat tyto vlastnosti 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 245.

- Volitelné: V případě potřeby nakonfigurujte adaptér prostředků tak, aby protokoloval varovné zprávy do standardního výstupního protokolu aplikačního serveru.

  Protokoly, varování a chybové zprávy adaptéru prostředků používají stejný mechanizmus jako 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 zprávy přejdou do souboru s názvem `mqjms.log`. Chcete-li konfigurovat adaptér prostředků tak, aby dále protokoloval varovné zprávy do standardního výstupního protokolu aplikačního serveru, nastavte pro aplikační server následující systémovou vlastnost prostředí JVM:

  ```
  -Dcom.ibm.msg.client.commonservices.log.outputName=mqjms.log,stdout
  ```

  Jedná se o stejnou vlastnost jako vlastnost, která se používá k řízení trasování pro IBM MQ classes for JMS. Stejně jako u IBM MQ classes for JMSje možné použít systémovou vlastnost ukazující na soubor `jms.config` (viz “Konfigurační soubor IBM MQ classes for JMS/Jakarta Messaging” na stránce 95). Informace o nastavení systémové vlastnosti prostředí JVM naleznete v dokumentaci k aplikačnímu serveru.

Konfigurace adaptéru prostředků pro spuštění ověřovacího testu instalace

- Nakonfigurujte adaptér prostředků pro spuštění programu IVT (Installation Verification Test) dodaného s adaptérem prostředků IBM MQ .

Chcete-li získat informace o tom, co musíte nakonfigurovat, abyste mohli spustit program IVT, prohlédněte si téma “Ověření instalace adaptéru prostředků” na stránce 476.

To je důležité, protože služba IBM může vyžadovat spuštění tohoto programu, aby označila, že jakýkoli jiný aplikační server nežIBM byl správně nakonfigurován.

Důležité: Před spuštěním programu musíte nakonfigurovat adaptér prostředků.

Konfigurace pro vlastnosti objektu ResourceAdapter

Objekt ResourceAdapter zapouzdřuje globální vlastnosti adaptéru prostředků IBM MQ , například úroveň diagnostického trasování. Chcete-li definovat tyto vlastnosti, použijte prostředky adaptéru prostředků, jak je popsáno v dokumentaci dodané s aplikačním serverem.

Objekt ResourceAdapter má dvě sady vlastností:

- Vlastnosti přidružené k diagnostickému trasování
- Vlastnosti přidružené k fondu připojení spravovanému adaptérem prostředků

Způsob, jakým tyto vlastnosti definujete, závisí na rozhraních administrace, která poskytuje váš aplikační server. Pokud používáte produkt WebSphere Application Server traditional, prohlédněte si téma “Konfigurace produktu WebSphere Application Server traditional” na stránce 438 nebo pokud používáte produkt WebSphere Liberty, viz “Konfigurace produktu WebSphere Liberty” na stránce 439. Další aplikační servery naleznete v dokumentaci k produktu pro váš aplikační server.

Další informace o definování vlastností přidružených k diagnostickému trasování naleznete v tématu Trasování IBM MQ adaptéru prostředků .

Adaptér prostředků spravuje interní fond připojení JMS připojení, která se používají k doručování zpráv do objektů MDB. Tabulka 63 na stránce 437 uvádí vlastnosti objektu ResourceAdapter , které jsou přidruženy k fondu připojení.

Tabulka 63. Vlastnosti objektu ResourceAdapter , které jsou přidruženy k fondu připojení

Název majetku	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 pro sdílení připojení JMS . Sdílení připojení není možné a tato vlastnost má vždy hodnotu 1.
reconnectionRetryPočet	Řetězec	5	Maximální počet pokusů adaptéru prostředků o opětovné připojení ke správci front IBM MQ v případě selhání připojení.
reconnectionRetryinterval	Řetězec	300 000	Doba v milisekundách, po kterou adaptér prostředků čeká před pokusem o opětovné připojení ke správci front IBM MQ .
Počet startupRetry	Řetězec	0	Výchozí počet pokusů o připojení MDB při spuštění, pokud není správce front spuštěn při spuštění aplikačního serveru.
startupRetryinterval	Řetězec	30 000	Výchozí doba spánku mezi pokusy o spuštění připojení (v milisekundách).
supportMQExtensions	Řetězec	ne	Vrátí chování produktu IBM MQ JMS zpět na chování předJMS 2.0 . Další informace viz “Vlastnost SupportMQExtensions” na stránce 317.

Tabulka 63. Vlastnosti objektu ResourceAdapter , které jsou přidruženy k fondu připojení (pokračování)			
Název majetku	**Typ**	**Výchozí hodnota**	**Popis**
nativeLibraryCesta	Řetězec	<empty>	Cesta, která má být použita k načtení knihovny IBM MQ JNI pro povolení připojení v režimu vazeb. Windows V systému Windows musí systémová cesta také obsahovat umístění odpovídající instalace produktu IBM MQ .

Je-li na aplikačním serveru implementován objekt MDB, vytvoří se nové připojení JMS a spustí se konverzace se správcem front za předpokladu, že není překročen maximální počet připojení určený vlastností maxConnection . Maximální počet objektů MDB se proto rovná maximálnímu počtu připojení. Pokud počet implementovaných objektů typu message-driven bean dosáhne tohoto maxima, jakýkoli pokus o implementaci jiného objektu typu message-driven bean se nezdaří. Je-li objekt typu message-driven bean zastaven, může jeho připojení použít jiný objekt typu message-driven bean.

Obecně platí, že má-li být implementováno mnoho objektů MDB, musíte zvýšit hodnotu vlastnosti maxConnections .

Vlastnosti reconnectionRetryPočet a reconnectionRetryřídí chování adaptéru prostředků při selhání připojení ke správci front IBM MQ , například kvůli selhání sítě. Dojde-li k selhání připojení, pozastaví adaptér prostředků doručení zpráv do všech objektů MDB dodaných tímto připojením na interval určený vlastností intervalu reconnectionRetry. Adaptér prostředků se poté pokusí znovu připojit ke správci front. Pokud se pokus nezdaří, adaptér prostředku provede další pokusy o opětovné připojení v intervalech určených vlastností intervalu reconnectionRetry, dokud nebude dosaženo limitu stanoveného vlastností reconnectionRetry. Pokud se všechny pokusy nezdaří, je doručení trvale zastaveno, dokud nejsou objekty MDB ručně restartovány.

Obecně platí, že objekt ResourceAdapter nevyžaduje žádnou administraci. Chcete-li však například povolit diagnostické trasování na systémech AIX and Linux , můžete nastavit následující vlastnosti:

```
traceEnabled:    true
traceLevel:      10
```

Tyto vlastnosti nemají žádný vliv, pokud nebyl adaptér prostředků spuštěn, což je například případ, kdy jsou aplikace používající prostředky IBM MQ spuštěny pouze v kontejneru klienta. V této situaci můžete nastavit vlastnosti pro diagnostické trasování jako systémové vlastnosti prostředí JVM (Java Virtual Machine). Vlastnosti můžete nastavit pomocí příznaku -D v příkazu **java** , jako v následujícím příkladu:

```
java ... -DtraceEnabled=true -DtraceLevel=6
```

Není třeba definovat všechny vlastnosti objektu ResourceAdapter . Všechny vlastnosti, které zůstaly nespecifikované, mají své výchozí hodnoty. Ve spravovaném prostředí je lepší nemíchat dva způsoby zadávání vlastností. Pokud je smícháte, systémové vlastnosti prostředí JVM budou mít 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 v rámci panelu vlastností adaptéru prostředků (viz Nastavení poskytovatele JMS v dokumentaci k produktu WebSphere Application Server traditional . Trasování je řízeno sekcí diagnostiky konfigurace WebSphere Application Server traditional . Další informace naleznete v tématu Práce s poskytovateli diagnostiky v dokumentaci k produktu WebSphere Application Server traditional .

Konfigurace produktu WebSphere Liberty

Adaptér prostředků je konfigurován pomocí prvků XML v souboru `server.xml` , jak ukazuje následující příklad:

JM 3.0

```
<featureManager>
...
    <feature>messaging-3.0</feature>
...
</featureManager>
    <variable name="wmqJmsClient.rar.location"
        value="F:/_rtc_wmq8005/_build/ship/lib/jca/wmq.jakarta.jmsra.rar"/>
...
        <wmqJmsClient supportMQExtensions="true" logWriterEnabled="true"/>
```

JMS 2.0

```
<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 konfigurovat příchozí komunikaci, definujte vlastnosti jednoho nebo více objektů ActivationSpec .

Vlastnosti objektu ActivationSpec určují, jak objekt typu message-driven bean (MDB) přijímá zprávy JMS z fronty IBM MQ . Transakční chování objektu MDB je definováno v deskriptoru implementace.

Objekt ActivationSpec má dvě sady vlastností:

- Vlastnosti, které se používají k vytvoření připojení JMS ke správci front IBM MQ
- Vlastnosti použité k vytvoření spotřebitele připojení JMS , který doručuje zprávy asynchronně při doručení do určené fronty.

Způsob, jakým definujete vlastnosti objektu ActivationSpec , závisí na rozhraních administrace, která poskytuje váš aplikační server.

Vlastnosti použité k vytvoření připojení JMS ke správci front IBM MQ

Všechny vlastnosti v souboru Tabulka 64 na stránce 440 jsou volitelné.

Tabulka 64. Vlastnosti objektu ActivationSpec , které se používají k vytvoření připojení JMS

Název majetku	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
applicationName	Řetězec	• Název vyvolávající třídy, je-li k dispozici, upravený tak, aby neobsahoval více než 28 znaků. Pokud není k dispozici, použije se řetězec `WebSphere MQ Klient pro jazyk Java`.	Název, pod kterým je aplikace registrována ve správci front. Tento název aplikace je zobrazen příkazem **`DISPLAY CONN MQSC/PCF`** (kde se pole nazývá **`APPLTAG`**). nebo na obrazovce IBM MQ Explorer **Připojení aplikace** (kde se pole nazývá **`App name`**).
brokerCCDurSubQueue [1]	Řetězec	• **SYSTEM.JMS.D.CC.SUBSCRIBER.QUEUE** • Název fronty	Název fronty, ze které spotřebitel připojení přijímá zprávy trvalého odběru
brokerCCSubFronta [1]	Řetězec	• **SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE** • Název fronty	Název fronty, ze které spotřebitel připojení přijímá zprávy dočasného odběru
brokerControlFronta [1]	Řetězec	• **SYSTEM.BROKER.CONTROL.QUEUE** • Název fronty	Název fronty řízení 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, ze které spotřebitel přechodných zpráv přijímá zprávy
brokerVersion [1]	Řetězec	• **neurčeno** -Po migraci zprostředkovatele z verze V6 na verzi V7nastavte tuto vlastnost tak, aby se záhlaví RFH2 již nepoužívala. Po migraci již tato vlastnost není relevantní. • **V1** -Chcete-li použít zprostředkovatele IBM MQ publish/subscribe broker.This je výchozí hodnota, pokud je TRANSPORT nastaven na BIND nebo CLIENT. • **V2** -Chcete-li použít zprostředkovatele IBM Integration Bus v nativním režimu. Tato hodnota je výchozí hodnota, pokud je hodnota TRANSPORT nastavena na DIRECT nebo DIRECTHTTP.	Verze používaného zprostředkovatele
ccdtURL	Řetězec	• **null** • Jednotný lokátor prostředků (URL)	URL , která identifikuje název a umístění souboru obsahujícího tabulku CCDT (Client Channel Definition Table) a určuje způsob přístupu k souboru.

Tabulka 64. Vlastnosti objektu ActivationSpec , které se používají k vytvoření připojení JMS (pokračování)			
Název majetku	**Typ**	**Platné hodnoty (výchozí hodnota tučně)**	**Popis**
CCSID	Řetězec	• **819** • Identifikátor kódované znakové sady podporovaný prostředím JVM (Java Virtual Machine).	Identifikátor kódované znakové sady 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 vyčištění publikování/odběru
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	• **DISABLED** -V daném okamžiku může být spuštěna pouze jedna instance trvalého odběratele tématu. • POVOLENO-Dvě nebo více instancí stejného trvalého odběratele tématu lze spustit současně, ale každá instance musí být spuštěna v samostatném prostředí JVM (Java Virtual Machine).	Zda mohou být dvě nebo více instancí stejného trvalého odběratele tématu spuštěny současně

Tabulka 64. Vlastnosti objektu ActivationSpec , které se používají k vytvoření připojení JMS (pokračování)

Název majetku	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
Vyhledávání connectionFactory	Řetězec	• **null** • Název rozhraní JNDI pro objekt ConnectionFactory	Je-li tato vlastnost nastavena, ActivationSpec vyhledá objekt JMS ConnectionFactory s uvedeným názvem rozhraní JNDI v oboru názvů rozhraní JNDI aplikačního serveru a poté použije vlastnosti tohoto objektu k vytvoření připojení JMS ke správci front IBM MQ s jednou výjimkou. Jedinou vlastností ActivationSpec , která se použije při vytváření připojení JMS , je clientID. Další informace viz “Vlastnosti ActivationSpec connectionFactoryLookup a destinationLookup” na stránce 453.
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 používaných pro příchozí komunikace. Je-li zadána volba **connectionNameList** , nahradí vlastnosti **hostname** a **port** . Tato vlastnost se používá k opětovnému připojení ke správcům front s více instancemi. **connectionNameList** má podobný tvar jako **localAddress**, ale nesmí být s ním zaměňován. **localAddress** uvádí charakteristiku lokální komunikace, zatímco **connectionNameList** uvádí, jak dosáhnout vzdáleného správce front.
V 9.3.0 V 9.3.0 dynamicallyBalanced [4]	Logická hodnota	• **ne** • ano	Určuje, zda může být tento objekt typu message-driven bean požádán o příjem zpráv z jiného správce front v rámci vyrovnávání aplikací v jednotném klastru.

Tabulka 64. Vlastnosti objektu ActivationSpec , které se používají k vytvoření připojení JMS (pokračování)

Název majetku	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
FAILIFQUIESCE	Logická hodnota	• **ano** • ne	Zda se volání určitých metod nezdaří, pokud se správce front nachází ve stavu uvedení do klidového stavu.
headerCompression	Řetězec	• **Není** • SYSTEM-Komprese záhlaví zprávy RLE je provedena	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, ve kterém je správce front umístěn. Vlastnosti **hostname** a **port** jsou po zadání nahrazeny vlastností **connectionNameList** .
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 závorky označují volitelnou komponentu	Pro připojení ke správci front tato vlastnost určuje jednu nebo obě následující věci: • Lokální síťové rozhraní, které se má použít • Lokální port nebo rozsah lokálních portů, které se mají použít **localAddress** má podobný tvar jako **connectionNameList**, ale nesmí být s ním zaměňován. **localAddress** uvádí charakteristiku lokální komunikace, zatímco **connectionNameList** uvádí, jak dosáhnout vzdáleného správce front.
messageCompression	Řetězec	• **Není** • Seznam jedné nebo více následujících hodnot oddělených prázdnými znaky: RLE ZLIBFAST ZLIBHIGH	Seznam technik, které lze použít pro kompresi dat zprávy na připojení

Tabulka 64. Vlastnosti objektu ActivationSpec , které se používají k vytvoření připojení JMS (pokračování)

Název majetku	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
messageRetention [1]	Logická hodnota	• **true** -Nepožadované zprávy zůstávají ve vstupní frontě. • false-Nežádoucí zprávy jsou řešeny podle jejich dispozičních možností	Zda spotřebitel připojení uchovává nežádoucí zprávy ve vstupní frontě
messageSelection [1]	Řetězec	• **client** • BROKER	Určuje, zda výběr zpráv provádí IBM MQ classes for JMS nebo zprostředkovatel. Výběr zprávy zprostředkovatelem není podporován, pokud má brokerVersion 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 modulů listener pro zprávy v relaci není k dispozici žádná vhodná zpráva, zvažte zvýšení hodnoty této vlastnosti. Tato vlastnost je relevantní pouze v případě, že má volba TRANSPORT hodnotu BIND nebo CLIENT.
Port	celé číslo	• **1414** • Číslo portu TCP	Port, na kterém správce front naslouchá. Vlastnosti **hostname** a **port** jsou po zadání nahrazeny vlastností **connectionNameList** .

Tabulka 64. Vlastnosti objektu ActivationSpec , které se používají k vytvoření připojení JMS (pokračování)

Název majetku	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
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 nebo rovné nule.	Verze, vydání, úroveň modifikace a opravná sada správce front, ke kterému se má MDB připojit.
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 je úplný název třídy, která implementuje rozhraní IBM MQ classes for Java , MQReceiveExit .	Identifikuje uživatelský program pro příjem kanálu nebo posloupnost uživatelských procedur pro příjem, které mají být spuštěny v posloupnosti.
receiveExitInit	Řetězec	• **null** • Řetězec obsahující jednu nebo více položek uživatelských dat oddělených čárkami	Uživatelská data, která jsou při volání předána uživatelským proceduře pro příjem kanálu.

Tabulka 64. Vlastnosti objektu ActivationSpec , které se používají k vytvoření připojení JMS (pokračování)			
Název majetku	**Typ**	**Platné hodnoty (výchozí hodnota tučně)**	**Popis**
rescanInterval [1]	celé číslo	• **5000** • Libovolné kladné celé číslo	Když spotřebitel zpráv v doméně typu point-to-point používá selektor zpráv k výběru zpráv, které chce přijmout, produkt IBM MQ classes for JMS vyhledá ve frontě IBM MQ vhodné zprávy v pořadí určeném atributem **MsgDeliverySequence** fronty. Když 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 její aktuální pozice ve frontě. Produkt IBM MQ classes for JMS pokračuje v prohledávání fronty tímto způsobem, dokud nedosáhne konce fronty nebo dokud nevyprší časový interval v milisekundách určený hodnotou této vlastnosti. V každém případě se produkt IBM MQ classes for JMS vrátí na začátek fronty, aby mohl pokračovat 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
securityExitInit	Řetězec	• **null** • Řetězec uživatelských dat	Uživatelská data, která jsou při volání předána programu uživatelské procedury pro zabezpečení zprávy kanálu
sendExit [3]	Řetězec	• **null** • Řetězec obsahující jednu nebo více položek oddělených čárkami, kde každá položka je úplný název třídy, která implementuje rozhraní IBM MQ classes for Java , MQSendExit .	Identifikuje uživatelský program odeslání kanálu nebo posloupnost uživatelských procedur odeslání, které mají být spuštěny v posloupnosti.

Tabulka 64. Vlastnosti objektu ActivationSpec , které se používají k vytvoření připojení JMS (pokračování)

Název majetku	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
SENDEXITINIT	Řetězec	• **null** • Řetězec obsahující jednu nebo více položek uživatelských dat oddělených čárkami	Uživatelská data, která se předávají uživatelským programům odeslání kanálu, když se volají
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 jinými připojeními JMS nejvyšší úrovně ze stejného procesu ke stejnému správci front, pokud se definice kanálu shodují.
sparseSubscriptions [1]	Logická hodnota	• **false** -Odběry přijímají často odpovídající zprávy. • true-Odběry přijímají zřídka odpovídající zprávy. Tato hodnota vyžaduje, aby frontu odběrů bylo možné otevřít pro procházení.	Řídí zásadu načítání zpráv objektu TopicSubscriber .
sslCertProdejny	Řetězec	• **null** • Řetězec jedné nebo více adres URL LDAP oddělených mezerami. Každá URL LDAP má formát: `ldap://host_name [: port ]` kde *název_hostitele* je název hostitele nebo adresa IP, *port* je číslo portu TCP a závorky označují volitelnou komponentu.	Servery LDAP (Lightweight Directory Access Protocol), které obsahují seznamy odvolaných certifikátů (CRL) pro použití v připojení TLS
SSLCIPHERSUITE	Řetězec	• **null** • Název CipherSuite	CipherSuite , která má být použita pro připojení TLS.
sslFipsVyžadováno [2]	Logická hodnota	• **ne** • ano	Zda musí připojení TLS používat sadu CipherSuite , kterou podporuje poskytovatel IBM Java JSSE FIPS (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 poskytnutém správcem front.
SSLRESETCOUNT	celé číslo	• **0** • Celé číslo v rozsahu 0-999 999 999 999	Celkový počet bajtů odeslaných a přijatých připojením TLS, než budou znovu vyjednány tajné klíče použité TLS

Tabulka 64. Vlastnosti objektu ActivationSpec , které se používají k vytvoření připojení JMS (pokračování)

Název majetku	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
Továrna sslSocket	Řetězec	Řetězec představující úplný název třídy poskytující implementaci rozhraní javax.net.ssl.SSLSocketFactory . Volitelně včetně argumentu, který má být předán metodě konstruktoru, uzavřeného v závorkách.	Všechna připojení zavedená v oboru spravovaného objektu používají sokety získané z této implementace rozhraní SSLSocketFactory .
statusRefreshInterval [1]	celé číslo	• **60000** • Libovolné kladné celé číslo	Interval (v milisekundách) mezi aktualizacemi dlouho běžící transakce, která zjistí, když odběratel ztratí připojení ke správci front. Tato vlastnost je relevantní pouze v případě, že vlastnost **subscriptionStore** má hodnotu QUEUE.
subscriptionStore [1]	Řetězec	• **BROKER** • MIGRATE • QUEUE	Určuje, kam produkt IBM MQ classes for JMS ukládá trvalá data o aktivních odběrech.

Tabulka 64. Vlastnosti objektu ActivationSpec , které se používají k vytvoření připojení JMS (pokračování)			
Název majetku	**Typ**	**Platné hodnoty (výchozí hodnota tučně)**	**Popis**
transportType	Řetězec	• **client** • Vazby • BINDINGS_THEN_CLIENT	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 tento pokus o připojení nezdaří, adaptér prostředků se pokusí vytvořit připojení v režimu klienta. z/OS Pokud byla specifikace aktivace spuštěná v systému WebSphere Application Server for z/OS konfigurována tak, aby používala režim transportu `BINDINGS_THEN_CLIENT` , a dříve navázané připojení je přerušeno, pak se všechny pokusy o opětovné připojení podle specifikace aktivace nejprve pokusí použít režim transportu `BINDINGS` . Pokud je pokus o připojení v režimu transportu `BINDINGS` neúspěšný, 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-rozpoznává pouze znakové zástupné znaky, jak jsou použity ve zprostředkovateli verze 1 • **TOPIC** -Uznává pouze zástupné znaky na úrovni tématu, které se používají ve zprostředkovateli verze 2.	Která verze syntaxe zástupných znaků se má použít

Notes:

1. Tuto vlastnost lze použít s verzí 70 IBM MQ classes for JMS.
2. Důležité informace o použití vlastnosti sslFipsRequired najdete v části “Omezení adaptéru prostředků IBM MQ” na stránce 427.

3. Informace o tom, jak nakonfigurovat adaptér prostředků tak, aby mohl vyhledat uživatelskou proceduru, naleznete v části “Konfigurace IBM MQ classes for JMS pro použití uživatelských procedur kanálu” na stránce 272.
4. V 9.3.0 Vlastnost dynamicallyBalanced není podporována ve spojení s podporou transakcí XA. Má-li parametr dynamicallyBalanced hodnotu "true", musí být objekt MDB konfigurován tak, aby zakázal transakce XA.

Vlastnosti použité k vytvoření spotřebitele připojení JMS

Poznámka: `destination` a **`destinationType`** musí být definovány explicitně. Všechny ostatní vlastnosti v souboru Tabulka 65 na stránce 450 jsou volitelné.

Tabulka 65. Vlastnosti objektu ActivationSpec , které se používají k vytvoření spotřebitele připojení JMS

Název majetku	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
cíl	Řetězec	Název místa určení	Cíl, ze kterého mají být přijímány zprávy. Vlastnost **`useJNDI`** určuje způsob interpretace hodnoty této vlastnosti.
destinationLookup	Řetězec	• **null** • Název rozhraní JNDI pro cílový objekt	Je-li tato vlastnost nastavena, ActivationSpec vyhledá objekt cíle JMS s určeným názvem rozhraní JNDI v oboru názvů rozhraní JNDI aplikačního serveru a poté použije vlastnosti tohoto objektu k vytvoření spotřebitele připojení JMS namísto ostatních vlastností určených v objektu ActivationSpec. Další informace viz téma “Vlastnosti ActivationSpec connectionFactoryLookup a destinationLookup” na stránce 453.
destinationType	Řetězec	• V 9.3.0 V 9.3.0 jakarta.jms.Queue (Jakarta Messaging 3.0) • V 9.3.0 V 9.3.0 jakarta.jms.Topic (Jakarta Messaging 3.0) • javax.jms.Queue (JMS 2.0) • javax.jms.Topic (JMS 2.0)	Typ cíle, fronty nebo tématu
maxMessages	celé číslo	• **1** • Kladné celé číslo	Maximální počet zpráv, které lze přiřadit k relaci serveru najednou. Pokud specifikace aktivace doručuje zprávy do objektu MDB v transakci 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, které používá spotřebitel 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 majetku	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
nonASFTimeout	celé číslo	• **0** • Kladné celé číslo	Kladná hodnota označuje, že je použito doručení mimo ASF. Hodnota je doba v milisekundách, po kterou požadavek na získání čeká na zprávy, které ještě nedorazily (volání get with wait). Výchozí hodnota 0 označuje, že je použito doručení ASF. Tento parametr je platný, pokud: • Aplikace běží na systému WebSphere Application Server 7.0 nebo novějším. • Aplikace je spuštěna v produktu WebSphere Liberty s použitím příslušné úrovně funkce klienta wmqJms. Další informace viz téma “Liberty a adaptér prostředků IBM MQ” na stránce 428.
nonASFRollbackPovoleno	Logická hodnota	• **false** -Zpráva se spotřebuje i v případě, že objekt MDB selže. • true-Selhání v objektu MDB způsobí odvolání zprávy do fronty.	Zda se doručení zprávy nachází v synchronizačním bodu IBM MQ , pokud není MDB netransakční. Ignoruje se, pokud je objekt typu message-driven bean transakční, nebo pokud je parametr **nonASFTimeout** nastaven na hodnotu 0.
poolTimeout	celé číslo	• **300000** • Kladné celé číslo	Doba v milisekundách, po kterou je nepoužívaná relace serveru zadržena otevřená ve fondu relací serveru, než je zavřena kvůli nečinnosti
READAHEADALLOWED	celé číslo	• **DESTINATION** -Určete, zda je povoleno dopředné čtení, pomocí odkazu na definici fronty nebo tématu. • DISABLED-Dopředné čtení není povoleno. • POVOLENO-Dopředné čtení je povoleno. • QUEUE-Určete, zda je dopředné čtení povoleno odkazem na definici fronty. • TOPIC-Určete, zda je dopředné čtení povoleno s odkazem na definici tématu.	Určuje, zda může podproces procházení specifikace aktivace použít dopředné čtení k procházení více zpráv z místa určení do interní vyrovnávací paměti před předáním relacím serveru k destruktivní spotřebě. **Poznámka:** Povolení dopředného čtení může mít za následek zvýšení počtu zpráv JMSCC0108 nebo snížení výkonu, nebo obojí, pokud rychlost zpracování MDB nemůže držet krok s rychlostí procházení zpráv z místa určení.

Tabulka 65. Vlastnosti objektu ActivationSpec , které se používají k vytvoření spotřebitele připojení JMS (pokračování)

Název majetku	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
readAheadClosePolicy	celé číslo	• **ALL** -Všechny zprávy ve vyrovnávací paměti dopředného interního čtení jsou před zastavením doručeny do objektu MDB. • CURRENT-Je dokončeno pouze aktuální vyvolání MDB, což může zanechávat zprávy v interní vyrovnávací paměti dopředného čtení, které jsou pak vyřazeny.	Co se stane se zprávami v interní vyrovnávací paměti dopředného čtení, když je objekt MDB zastaven administrátorem.
receiveCCSID	celé číslo	• **0** -Použít prostředí JVM `Charset.defaultCharset` • 1208- UTF-8 • Podporovaný identifikátor kódované znakové sady	Cílová vlastnost, která nastavuje cílový CCSID pro převod zpráv správce front. Hodnota je ignorována, pokud není parametr **`receiveConversion`** nastaven na hodnotu `QMGR`.
receiveConversion	Řetězec	• **CLIENT_MSG** • QMGR	Cílová vlastnost, která určuje, zda má správce front provést převod dat.
sharedSubscription	Logická hodnota	• **False** -objekt MDB by neměl otevírat odběr jako sdílený odběr. • True-objekt MDB by měl otevřít odběr jako sdílený odběr (s pravidly, která JMS 2.0 implikuje, viz specifikace JMS 2.0 na adrese Java.net).	Určuje, jak je objekt MDB řízen ze sdíleného odběru. Další informace o použití této vlastnosti viz "Příklady, jak definovat vlastnost sharedSubscription" na stránce 456.
startTimeout	celé číslo	• **10 000** • Kladné celé číslo	Doba v milisekundách, během které musí být doručení zprávy do objektu MDB zahájeno po naplánování práce na doručení zprávy. Pokud tato doba uplyne, zpráva se vrátí zpět do fronty.
subscriptionDurability	Řetězec	• **NonDurable** -Přechodný odběr se používá k doručování zpráv do objektu MDB odebírajícího dané téma. • Trvalý-trvalý odběr se používá k doručení zpráv do objektu MDB přihlášeného k odběru tématu.	Zda se k doručení zpráv do objektu MDB přihlášeného k odběru tématu používá trvalý nebo přechodný 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 majetku	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 jednoho z následujících objektů v oboru názvů JNDI aplikačního serveru: – V 9.3.0 V 9.3.0 jakarta.jms.Queue (Jakarta Messaging 3.0) – V 9.3.0 V 9.3.0 jakarta.jms.Topic (Jakarta Messaging 3.0) – javax.jms.Queue (JMS 2.0) – javax.jms.Topic (JMS 2.0)	Deprecated Určuje způsob interpretace hodnoty vlastnosti s názvem destination **Poznámka:** Tato vlastnost je v produktu IBM MQ 9.0zamítnuta. Místo toho by měla být použita vlastnost destinationLookup .

Konflikty vlastností a závislosti

Objekt ActivationSpec může mít konfliktní vlastnosti. Můžete například určit vlastnosti TLS pro připojení v režimu vazeb. V tomto případě je chování určeno typem přenosu a doménou systému zpráv, která je buď dvoubodová, nebo publikování/odběr, jak určuje vlastnost **destinationType** . Všechny vlastnosti, které nelze použít pro zadaný typ transportu nebo doménu systému zpráv, jsou ignorovány.

Pokud definujete vlastnost, která vyžaduje definování jiných vlastností, ale tyto další vlastnosti nedefinujete, objekt ActivationSpec vygeneruje výjimku InvalidPropertypři volání metody validate () během implementace objektu MDB. Výjimka je nahlášena administrátorovi aplikačního serveru způsobem, který závisí na aplikačním serveru. Pokud například nastavíte vlastnost subscriptionDurability na hodnotu Durable, což znamená, že chcete používat trvalé odběry, musíte také definovat vlastnost **subscriptionName** .

Pokud jsou definovány vlastnosti s názvem **ccdtURL** a **channel** , dojde k výjimce InvalidProperty. Pokud však definujete pouze vlastnost **ccdtURL** , ponecháte vlastnost s názvem **channel** s výchozí hodnotou SYSTEM.DEF.SVRCONNnení vyvolána žádná výjimka a tabulka definic kanálů klienta identifikovaná vlastností **ccdtURL** se používá ke spuštění připojení JMS .

Vlastnosti ActivationSpec connectionFactoryLookup a destinationLookup

Specifikace JMS 2.0 zavedla dvě nové vlastnosti ActivationSpec . Vlastnosti connectionFactoryLookup a destinationLookup lze poskytnout s názvem JNDI spravovaného objektu, který má být použit jako předvolba k ostatním vlastnostem ActivationSpec .

Pokud je například továrna připojení definována v souboru JNDI a název JNDI daného objektu je určen ve vlastnosti vyhledávání connectionFactorypro specifikaci aktivace, budou všechny vlastnosti továrny připojení definované v souboru JNDI použity jako předvolba vlastností v souboru Tabulka 64 na stránce 440.

Pokud je cíl definován v souboru JNDI a název JNDI je nastaven ve vlastnosti ActivationSpec' s destinationLookup , pak se hodnoty, které se používají jako předvolba k hodnotám v souboru Tabulka

65 na stránce 450. Další informace o použití těchto dvou vlastností viz “Vlastnosti ActivationSpec connectionFactoryLookup a destinationLookup” na stránce 453.

Tyto dvě vlastnosti lze použít k určení JNDI názvů ConnectionFactory a cílových objektů, které se používají jako předvolba k vlastnostem ActivationSpec , jak je definováno v Tabulka 64 na stránce 440 a Tabulka 65 na stránce 450.

Je důležité si uvědomit následující body, které podrobně popisují, jak tyto vlastnosti fungují.

Vyhledávání connectionFactory
ConnectionFactory , která se vyhledává z JNDI , se používá jako zdroj vlastností uvedených v Tabulka 64 na stránce 440. Objekt ConnectionFactory se nepoužívá pro skutečné vytvoření žádných připojení JMS . Jsou dotazovány pouze vlastnosti objektu. Tyto vlastnosti z ConnectionFactory přepíší všechny vlastnosti, které jsou definovány v objektu ActivationSpec. Jedná se o jedinou výjimku. Má-li vlastnost ActivationSpec nastavenou vlastnost **ClientID** , pak hodnota této vlastnosti přepíše hodnotu zadanou v ConnectionFactory. Důvodem je skutečnost, že společný scénář používá jedinou továrnu připojení ConnectionFactory s více aktivními specifikacemi ActivationSpecs. To zjednodušuje administraci. Specifikace JMS 2.0 však uvádí, že každé připojení JMS vytvořené z ConnectionFactory by mělo mít jedinečný **ClientID**. Z tohoto důvodu musí mít ActivationSpecs možnost přepsat libovolnou hodnotu nastavenou na ConnectionFactory. Není-li v objektu ActivationSpecnastaven žádný parametr **ClientID** , bude použita libovolná hodnota továrny připojení.

destinationLookup
Vlastnosti **Destination** a **UseJndi** jsou definovány v objektu ActivationSpec. Je-li příznak **UseJndi** nastaven na hodnotu `true`, bude text určený ve vlastnosti místa určení považován za název JNDI a cílový objekt s tímto názvem JNDI bude vyhledán z adresáře JNDI.

Vlastnost destinationLookup se chová přesně stejným způsobem. Pokud byl nastaven, vyhledá se cílový objekt s názvem JNDI určeným vlastností z adresáře JNDI. Tato vlastnost má přednost před vlastností **useJNDI** .

Deprecated Vlastnost useJNDI je zamítnuta na adrese IBM MQ 9.0 , protože vlastnost **destinationLookup** je specifikací JMS 2.0 nebo pozdější ekvivalentní provedení stejné funkce.

Vlastnosti ActivationSpec bez ekvivalentů v IBM MQ classes for JMS

Většina vlastností objektu ActivationSpec je ekvivalentní vlastnostem objektů IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messaging nebo parametrům metod IBM MQ classes for JMS IBM MQ classes for Jakarta Messaging . Avšak tři vlastnosti ladění a jedna vlastnost použitelnosti nemají v IBM MQ classes for JMS nebo IBM MQ classes for Jakarta Messagingžádné ekvivalenty:

startTimeout
Doba v milisekundách, po kterou správce Work Manager aplikačního serveru čeká na dostupnost prostředků poté, co adaptér prostředků naplánuje pracovní objekt pro doručení zprávy do objektu MDB. Pokud tato doba uplyne před spuštěním doručení zprávy, dojde k vypršení časového limitu pracovního objektu, zpráva se vrátí zpět do fronty a adaptér prostředků se pak může pokusit zprávu znovu doručit. Do diagnostického trasování se zapíše varování, je-li povoleno, ale jinak neovlivní proces doručování zpráv. Můžete očekávat, že k tomuto stavu dojde pouze v době, kdy aplikační server vykazuje velmi vysokou zátěž. Pokud se podmínka vyskytuje pravidelně, zvažte zvýšení hodnoty této vlastnosti, abyste správci Work Manager poskytli delší dobu pro naplánování doručení zprávy.

Hloubka maxPool
Maximální počet relací serveru ve fondu relací serveru, které používá spotřebitel připojení. Při vytvoření relace serveru zahájí 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 velkého objemu, ale používá více prostředků aplikačního serveru. Má-li být implementováno mnoho objektů typu message-driven bean, zvažte možnost zmenšení hloubky fondu, abyste udrželi zátěž na aplikačním serveru na spravovatelné úrovni. Každý spotřebitel připojení používá svůj vlastní fond relací serveru, takže tato vlastnost nedefinuje celkový počet relací serveru, které jsou k dispozici pro všechny spotřebitele připojení.

poolTimeout
Doba v milisekundách, po kterou je nepoužívaná relace serveru zadržena otevřená ve fondu relací serveru, než je zavřena kvůli nečinnosti. Přechodné zvýšení pracovní zátěže zpráv způsobí vytvoření dalších relací serveru za účelem rozdělení zátěže, ale po návratu pracovní zátěže zpráv do normálního stavu zůstanou další relace serveru ve fondu a nebudou použity.

Při každém použití relace serveru je označena časovým razítkem. Podproces scavenger pravidelně kontroluje, zda byla každá relace serveru použita v období určeném touto vlastností. Pokud nebyla relace serveru použita, je uzavřena a odebrána z fondu relací serveru. Relace serveru nemusí být uzavřena okamžitě po uplynutí uvedené doby. Tato vlastnost představuje minimální dobu nečinnosti před odebráním.

useJNDI
Popis této vlastnosti viz Tabulka 65 na stránce 450.

Implementace objektu MDB

Chcete-li implementovat objekt MDB, nejprve definujte vlastnosti objektu ActivationSpec a zadejte vlastnosti, které objekt MDB vyžaduje. Následující příklad je typickou sadou vlastností, které můžete definovat explicitně:

V 9.3.0 JM 3.0 V 9.3.0

```
channel:           SYSTEM.DEF.SVRCONN
destination:       SYSTEM.DEFAULT.LOCAL.QUEUE
destinationType:   jakarta.jms.Queue
hostName:          192.168.0.42
messageSelector:   color='red'
port:              1414
queueManager:      ExampleQM
transportType:     CLIENT
```

JMS 2.0

```
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 vlastnosti k vytvoření objektu ActivationSpec , který je poté přidružen k objektu MDB. Vlastnosti objektu ActivationSpec určují způsob doručení zpráv do objektu MDB. Implementace objektu typu message-driven bean se nezdaří, pokud objekt typu message-driven bean vyžaduje distribuované transakce, ale adaptér prostředků nepodporuje distribuované transakce. Informace o instalaci adaptéru prostředků tak, aby byly podporovány distribuované transakce, naleznete v části “Instalace adaptéru prostředků IBM MQ” na stránce 431.

Pokud více než jeden objekt MDB přijímá zprávy ze stejného místa určení, je zpráva odeslaná v doméně typu point-to-point přijata pouze jedním objektem MDB, a to i v případě, že ostatní objekty MDB jsou vhodné pro přijetí zprávy. Pokud dva objekty typu message-driven bean používají různé selektory zpráv a příchozí zpráva odpovídá oběma selektorům zpráv, obdrží zprávu pouze jeden z objektů typu message-driven bean. Objekt typu message-driven bean, který byl vybrán pro příjem zprávy, není definován a nelze se spoléhat na konkrétní objekt typu message-driven bean, který zprávu přijímá. Zprávy odeslané v doméně publikování/odběru jsou přijímány všemi vhodnými MDB.

Za určitých okolností může být zpráva doručená do objektu MDB odvolána do fronty IBM MQ . K tomuto odvolání může dojít například v případě, že je zpráva doručena v rámci pracovní jednotky, která je poté odvolána. Zpráva, která je odvolána, je znovu doručena, ale chybně formátovaná zpráva může opakovaně způsobit selhání MDB, a proto nemůže být doručena. Taková zpráva se nazývá nezpracovatelnou zprávou. Můžete nakonfigurovat produkt IBM MQ tak, aby produkt IBM MQ classes for JMS automaticky přenesl nezpracovatelnou zprávu do jiné fronty pro další vyšetřování nebo vyřazení zprávy.

Podrobnosti o tom, jak zpracovat nezpracovatelné zprávy, viz “Zpracování nezpracovatelných zpráv v adresáři IBM MQ classes for JMS” na stránce 225.

Související pojmy

Určení, že se za běhu v klientu MQI používají pouze specifikace CipherSpecs s certifikací FIPS.

Standard FIPS (Federal Information Processing Standards) pro AIX, Linux, and Windows

Související úlohy

Konfigurace prostředků JMS v produktu WebSphere Application Server

Příklady, jak definovat vlastnost sharedSubscription

Můžete definovat vlastnost sharedSubscription specifikace aktivace v rámci souboru WebSphere Liberty `server.xml` . Případně můžete definovat vlastnost v rámci objektu typu message-driven bean (MDB) pomocí anotací.

Příklad: Definování v rámci souboru Liberty `server.xml`

V rámci souboru WebSphere Liberty `server.xml` definujete specifikaci aktivace, jak ukazuje následující příklad. Tento příklad vytvoří trvalý sdílený odběr pro správce front na lokálním hostiteli/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: Definice v rámci objektu MDB

Můžete také definovat vlastnost sharedSubscription v rámci objektu MDB pomocí anotací, jak ukazuje následující příklad:

```
@ActioncationConfigProperty(propertyName ="sharedSubscription",
propertyValue = "true")
```

Následující příklad ukazuje část kódu MDB, která používá metodu anotací:

V 9.3.0 JM 3.0 V 9.3.0

```
/**
 * Message-Driven Bean example using Annotations for configuration
 */
@MessageDriven(
  activationConfig = {
     @ActivationConfigProperty(
        propertyName = "destinationType", propertyValue = "jakarta.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
   }

}
```

JMS 2.0

```
/**
 * 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

Odběratelé a odběry

Trvanlivost předplatného

“Klonované a sdílené odběry” na stránce 315

V produktu IBM MQ 8.0 nebo novějším existují dvě metody, jak poskytnout více spotřebitelům přístup ke stejnému odběru. Tyto dvě metody používají klonované odběry nebo sdílené odběry.

Konfigurace adaptéru prostředků pro odchozí komunikaci

Chcete-li konfigurovat odchozí komunikaci, definujte vlastnosti objektu ConnectionFactory a spravovaného cílového objektu.

Příklad použití odchozí komunikace

Při použití odchozí komunikace aplikace spuštěná na aplikačním serveru spustí připojení ke správci front a poté odešle zprávy do svých front a přijme zprávy ze svých front synchronním způsobem. Například následující metoda servletu doGet() používá odchozí komunikaci:

V 9.3.0 JM 3.0 V 9.3.0

```
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 = (jakarta.jms.ConnectionFactory) ic.lookup("myCF");
     Queue q = (jakarta.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();
}
```

JMS 2.0

```
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 HTTP , načte objekt ConnectionFactory a objekt Queue z oboru názvů JNDI a použije tyto objekty k odeslání zprávy do fronty IBM MQ . Servlet poté obdrží zprávu, kterou odeslal.

Prostředky potřebné pro odchozí komunikaci

Chcete-li konfigurovat odchozí komunikaci, definujte prostředky Java EE Connector Architecture (JCA) v následujících kategoriích:

- Vlastnosti objektu ConnectionFactory, který aplikační server používá k vytvoření objektu JMS ConnectionFactory .
- Vlastnosti spravovaného cílového objektu, které aplikační server používá k vytvoření objektu JMS Fronta nebo JMS Téma.

Způsob, jakým tyto vlastnosti definujete, závisí na rozhraních administrace, která poskytuje váš aplikační server. Objekty ConnectionFactory, Queue a Topic vytvořené aplikačním serverem jsou vázány na obor názvů rozhraní JNDI, ze kterého mohou být načteny aplikací.

Obvykle definujete jeden objekt ConnectionFactory pro každého správce front, ke kterému se mohou aplikace připojovat. Definujete jeden objekt fronty pro každou frontu, ke které mohou aplikace potřebovat přístup v doméně typu point-to-point. A definujete jeden objekt tématu pro každé téma, které mohou aplikace chtít publikovat nebo odebírat. Objekt ConnectionFactory může být nezávislý na doméně. Alternativně může jít o objekt továrny QueueConnectionpro domény typu point-to-point nebo o objekt továrny TopicConnectionpro domény publikování/odběru.

Tip: V produktu JMS 2.0lze továrnu připojení použít k vytvoření připojení i kontextů. V důsledku toho je možné, aby byl fond připojení přidružen k továrně připojení, která obsahuje kombinaci připojení i kontextů. Doporučuje se, aby se továrna připojení používala pouze pro vytváření připojení nebo kontextů. Tím se

zajistí, že fond připojení pro danou továrnu připojení bude obsahovat pouze objekty jednoho typu, což tento fond zefektivní.

Vlastnosti objektu ConnectionFactory

Tabulka 66 na stránce 459 uvádí vlastnosti objektu ConnectionFactory . Aplikační server používá tyto vlastnosti k vytvoření objektu JMS ConnectionFactory .

Tabulka 66. Vlastnosti objektu ConnectionFactory

Název majetku	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
applicationName	Řetězec	• Název vyvolávající třídy, je-li k dispozici, upravený tak, aby neobsahoval více než 28 znaků. Pokud není k dispozici, použije se řetězec `WebSphere MQ Klient pro jazyk Java` .	Název, pod kterým je aplikace registrována ve správci front. Tento název aplikace je zobrazen pomocí příkazu **`DISPLAY CONN MQSC/PCF`** (kde se pole nazývá **`APPLTAG`**) nebo v zobrazení IBM MQ Průzkumník **Připojení aplikací** (kde se pole nazývá **`App name`**).
Fronta brokerCCSub	Řetězec	• **SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE** • Název fronty	Název fronty, z níž spotřebitel připojení přijímá zprávy dočasného odběru.
Fronta brokerControl	Řetězec	• **SYSTEM.BROKER.CONTROL.QUEUE** • Název fronty	Název fronty řízení zprostředkovatele.
Fronta brokerPub	Řetězec	• **SYSTEM.BROKER.DEFAULT.STREAM** • Název fronty	Název fronty, kam jsou odesílány publikované zprávy (fronta proudu).
Správce brokerQueue	Řetězec	• **"" (prázdný řetězec)** • Název správce front	Název správce front, v němž je zprostředkovatel spuštěn.
Fronta brokerSub	Řetězec	• **SYSTEM.JMS.ND.SUBSCRIBER.QUEUE** • Název fronty	Název fronty, ze které spotřebitel přechodných zpráv přijímá zprávy. Další informace viz vlastnost BROKERSUBQ .

Tabulka 66. Vlastnosti objektu ConnectionFactory (pokračování)

Název majetku	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
brokerVersion	Řetězec	• **neurčeno** -Po migraci zprostředkovatele z verze V6 na verzi V7nastavte tuto vlastnost tak, aby se záhlaví RFH2 již nepoužívala. Po migraci již tato vlastnost není relevantní. • **V1** -Chcete-li použít zprostředkovatele publikování/odběru IBM MQ . Tato hodnota je výchozí hodnota, pokud je hodnota TRANSPORT nastavena na BIND nebo CLIENT. • **V2** -Chcete-li použít zprostředkovatele IBM Integration Bus v nativním režimu. Tato hodnota je výchozí hodnota, pokud je hodnota TRANSPORT nastavena na DIRECT nebo DIRECTHTTP.	Verze používaného zprostředkovatele.
ccdtURL	Řetězec	• **null** • Jednotný lokátor prostředků (URL)	URL , která identifikuje název a umístění souboru obsahujícího tabulku CCDT (Client Channel Definition Table) a určuje způsob přístupu k souboru.
CCSID	Řetězec	• **819** • Identifikátor kódované znakové sady podporovaný prostředím JVM (Java Virtual Machine).	Identifikátor kódované znakové sady pro připojení.
kanál	Řetězec	• **SYSTEM.DEF.SVRCONN** • Název kanálu MQI	Název kanálu MQI, který má být použit.
cleanupInterval	celé číslo	• **3 600 000** • Kladné celé číslo	Interval, v milisekundách, mezi spuštěními na pozadí obslužného programu vyčištění publikování/odběru.
cleanupLevel	Řetězec	• **Bezpečný** • ŽÁDNÉ • velká • Vynutit • NONDUR	Úroveň vyčištění pro úložiště odběrů založené na zprostředkovateli.
clientID	Řetězec	• **null** • Identifikátor klienta	Identifikátor klienta pro účely připojení.

Tabulka 66. Vlastnosti objektu ConnectionFactory (pokračování)

Název majetku	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
cloneSupport	Řetězec	• **DISABLED** -V daném okamžiku může být spuštěna pouze jedna instance trvalého odběratele tématu. • POVOLENO-Dvě nebo více instancí stejného trvalého odběratele tématu lze spustit současně, ale každá instance musí být spuštěna v samostatném prostředí JVM (Java Virtual Machine).	Určuje, zda mohou být dvě nebo více instancí stejného trvalého odběratele tématu spuštěny současně.
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 používaných pro odchozí komunikaci. **connectionNameList** nahrazuje vlastnosti **hostname** a **port** . Tato vlastnost se používá k opětovnému připojení ke správcům front s více instancemi. **connectionNameList** má podobný tvar jako **localAddress**, ale nesmí být s ním zaměňován. **localAddress** uvádí charakteristiku lokální komunikace, zatímco **connectionNameList** uvádí, jak dosáhnout vzdáleného správce front.
FAILIFQUIESCE	Logická hodnota	• **ano** • ne	Určuje, zda se volání určitých metod nezdaří, pokud je správce front ve stavu uvedení do klidového stavu.
headerCompression	Řetězec	• **Není** • SYSTEM-Komprese záhlaví zprávy RLE je provedena.	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, ve kterém je správce front umístěn. Vlastnosti **hostname** a **port** jsou po zadání nahrazeny vlastností **connectionNameList** .

Tabulka 66. Vlastnosti objektu ConnectionFactory (pokračování)

Název majetku	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
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 závorky označují volitelnou komponentu	Pro připojení ke správci front tato vlastnost uvádí jednu nebo obě z následujících možností: • Lokální síťové rozhraní, které se má použít • Lokální port nebo rozsah lokálních portů, které se mají použít **localAddress** má podobný tvar jako **connectionNameList**, ale nesmí být s ním zaměňován. **localAddress** uvádí charakteristiku lokální komunikace, zatímco **connectionNameList** uvádí, jak dosáhnout vzdáleného správce front.
messageCompression	Řetězec	• **Není** • Seznam jedné nebo více následujících hodnot oddělených prázdnými znaky: RLE ZLIBFAST ZLIBHIGH	Seznam technik, které lze použít pro kompresi dat zprávy na připojení.
messageSelection	Řetězec	• **client** • BROKER	Určuje, zda výběr zpráv provádí IBM MQ classes for JMS nebo zprostředkovatel. Výběr zpráv zprostředkovatelem není podporován, pokud má parametr **brokerVersion** 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 majetku	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
pollingInterval	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 modulů listener pro zprávy v relaci není k dispozici žádná vhodná zpráva, zvažte zvýšení hodnoty této vlastnosti. Tato vlastnost je relevantní pouze v případě, že vlastnost **TRANSPORT** má hodnotu `BIND` nebo `CLIENT`.
Port	celé číslo	• **1414** • Číslo portu TCP	Port, na kterém správce front naslouchá. Vlastnosti **hostname** a **port** jsou po zadání nahrazeny vlastností **connectionNameList** .
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 nebo rovné nule.	Verze, vydání, úroveň modifikace a opravný balík správce front, ke kterému se tato aplikace hodlá připojovat.
pubAckInterval	celé číslo	• **25** • Kladné celé číslo	Počet zpráv publikovaných vydavatelem, než produkt IBM MQ classes for JMS vyžádá 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 je úplný název třídy, která implementuje rozhraní IBM MQ classes for Java , MQReceiveExit .	Identifikuje uživatelský program pro příjem kanálu nebo posloupnost uživatelských procedur pro příjem, které mají být spuštěny v posloupnosti.

Tabulka 66. Vlastnosti objektu ConnectionFactory (pokračování)

Název majetku	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
receiveExitInit	Řetězec	• **null** • Řetězec obsahující jednu nebo více položek uživatelských dat oddělených čárkami	Uživatelská data, která jsou při volání předána do programů uživatelské procedury pro příjem kanálu.
rescanInterval	celé číslo	• **5000** • Libovolné kladné celé číslo	Když spotřebitel zpráv v doméně typu point-to-point používá selektor zpráv k výběru zpráv, které chce přijmout, produkt IBM MQ classes for JMS vyhledá ve frontě IBM MQ vhodné zprávy v pořadí určeném atributem **MsgDeliverySequence** fronty. Když 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 její aktuální pozice ve frontě. Produkt IBM MQ classes for JMS pokračuje v prohledávání fronty tímto způsobem, dokud nedosáhne konce fronty nebo dokud nevyprší časový interval v milisekundách určený hodnotou této vlastnosti. V každém případě se produkt IBM MQ classes for JMS vrátí na začátek fronty, aby mohl pokračovat 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.
securityExitInit	Řetězec	• **null** • Řetězec uživatelských dat	Uživatelská data, která jsou při volání předána programu uživatelské procedury pro zabezpečení zprávy kanálu.
SENDCHECKCOUNT	celé číslo	• **0** • Libovolné kladné celé číslo	Počet volání odeslání, která mají být povolena mezi kontrolou asynchronních chyb vložení v rámci jedné netransakční relace JMS .

Tabulka 66. Vlastnosti objektu ConnectionFactory (pokračování)

Název majetku	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
sendExit [3]	Řetězec	• **null** • Řetězec obsahující jednu nebo více položek oddělených čárkami, kde každá položka je úplný název třídy, která implementuje rozhraní IBM MQ classes for Java , MQSendExit .	Identifikuje uživatelský program odeslání kanálu nebo posloupnost uživatelských procedur odeslání, které mají být spuštěny v posloupnosti.
SENDEXITINIT	Řetězec	• **null** • Řetězec obsahující jednu nebo více položek uživatelských dat oddělených čárkami	Uživatelská data, která jsou předána uživatelským programům odeslání kanálu při volání.
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 jinými připojeními JMS nejvyšší úrovně ze stejného procesu ke stejnému správci front, pokud se shodují definice kanálů.
sparseSubscriptions	Logická hodnota	• **false** -Odběry přijímají často odpovídající zprávy. • true-Odběry přijímají zřídka odpovídající zprávy. Tato hodnota vyžaduje, aby frontu odběrů bylo možné otevřít pro procházení.	Řídí zásadu načítání zpráv objektu TopicSubscriber .
sslCertProdejny	Řetězec	• **null** • Řetězec jedné nebo více adres URL LDAP oddělených mezerami. Každá URL LDAP má formát: `ldap://host_name [: port ]` kde *název_hostitele* je název hostitele nebo adresa IP, *port* je číslo portu TCP a závorky označují volitelnou komponentu.	Servery LDAP (Lightweight Directory Access Protocol), které obsahují seznamy odvolaných certifikátů (CRL) pro použití v připojení TLS.
SSLCIPHERSUITE	Řetězec	• **null** • Název CipherSuite	CipherSuite , která má být použita pro připojení TLS.
sslFipsVyžadováno [2]	Logická hodnota	• **ne** • ano	Zda musí připojení TLS používat sadu CipherSuite , která je podporována poskytovatelem FIPS JSSE 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 poskytnutém správcem front.

Tabulka 66. Vlastnosti objektu ConnectionFactory (pokračování)

Název majetku	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
SSLRESETCOUNT	celé číslo	• **0** • Celé číslo v rozsahu 0-999 999 999 999	Celkový počet bajtů odeslaných a přijatých připojením TLS, než budou znovu vyjednány tajné klíče použité TLS.
Továrna sslSocket	Řetězec	Řetězec představující úplný název třídy poskytující implementaci rozhraní javax.net.ssl.SSLSocketFactory , volitelně včetně argumentu, který má být předán metodě konstruktoru, uzavřený v závorkách.	Všechna připojení zavedená v oboru spravovaného cílového objektu používají sokety získané z této implementace rozhraní SSLSocketFactory .
statusRefreshInterval	celé číslo	• **60000** • Libovolné kladné celé číslo	Interval (v milisekundách) mezi aktualizacemi dlouho běžící transakce, která zjistí, když odběratel ztratí připojení ke správci front. Tato vlastnost je relevantní pouze v případě, že vlastnost **SUBSTORE** má hodnotu QUEUE.
subscriptionStore	Řetězec	• **BROKER** • MIGRATE • QUEUE	Určuje, kam produkt IBM MQ classes for JMS ukládá trvalá data o aktivních odběrech.
targetClientOdpovídající	Logická hodnota	• **ano** • ne	Zda má zpráva odpovědi odeslaná do fronty určené polem záhlaví JMSReplyTo příchozí zprávy záhlaví MQRFH2 pouze v případě, že příchozí zpráva obsahuje záhlaví MQRFH2 . Tuto vlastnost můžete také konfigurovat pro specifikaci aktivace. Další informace viz téma “Konfigurace vlastnosti targetClientOdpovídající vlastnosti pro specifikaci aktivace” na stránce 474.

Tabulka 66. Vlastnosti objektu ConnectionFactory (pokračování)

Název majetku	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 vytvořeny dočasné fronty JMS . Použít SYSTEM.DEFAULT.MODEL.QUEUE , pokud platí oba následující příkazy: • Vaše aplikace používá dočasnou frontu, která bude přijímat dočasné zprávy. • Dočasnou frontu ve správci front, na kterého odkazuje ConnectionFactory , vytvoří vždy pouze jedna aplikace. Všimněte si, že SYSTEM.DEFAULT.MODEL.QUEUE může v daném okamžiku otevřít pouze jedna aplikace. Použijte SYSTEM.JMS.TEMPQ.MODEL v následujících situacích: • Pokud vaše aplikace používá dočasnou frontu, která bude přijímat trvalé zprávy. • Pokud se ke správci front, na kterého odkazuje ConnectionFactory , může připojit více aplikací a tyto aplikace musí současně vytvářet dočasné fronty. Definujte novou modelovou frontu s atributem **DEFPSIST** nastaveným na hodnotu YESa atributem **DEFSOPT** nastaveným na hodnotu SHARED v následující situaci: • Pokud vaše aplikace používá dočasnou frontu, která bude přijímat dočasné zprávy, a více aplikací se připojí ke správci front, na kterého odkazuje ConnectionFactory , a tyto aplikace musí současně vytvářet dočasné fronty. Při vytvoření nové modelové fronty nastavte vlastnost **temporaryModel** na název nové modelové fronty.

Tabulka 66. Vlastnosti objektu ConnectionFactory (pokračování)

Název majetku	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
tempQPrefix	Řetězec	• **"" (prázdný řetězec)** • Předpona, kterou lze použít 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 neprázdný 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 IBM MQ .
TEMPTOPICPREFIX	Řetězec	Jakýkoli nenulový řetězec sestávající pouze z platných znaků pro řetězec tématu IBM MQ	Při vytváření dočasných témat produkt JMS vygeneruje řetězec tématu ve tvaru "TEMP/*TEMPTOPICPREFIX*/*unique_id* ", nebo pokud je tato vlastnost ponechána s výchozí hodnotou, pouze "TEMP/*unique_id*". Zadání neprázdného parametru **TEMPTOPICPREFIX** umožňuje definovat specifické modelové fronty pro vytváření spravovaných front pro odběratele dočasných témat vytvořených v rámci tohoto připojení.

Tabulka 66. Vlastnosti objektu ConnectionFactory (pokračování)

Název majetku	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
transportType	Řetězec	• **client** • Vazby • BINDINGS_THEN_CLIENT	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 tento pokus o připojení nezdaří, adaptér prostředků se pokusí vytvořit připojení v režimu klienta. z/OS Pokud byla specifikace aktivace spuštěná v systému WebSphere Application Server for z/OS konfigurována tak, aby používala režim transportu `BINDINGS_THEN_CLIENT`, a dříve navázané připojení je přerušeno, pak se všechny pokusy o opětovné připojení podle specifikace aktivace nejprve pokusí použít režim transportu `BINDINGS`. Pokud je pokus o připojení v režimu transportu `BINDINGS` neúspěšný, 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-rozpoznává pouze znakové zástupné znaky, jak jsou použity ve zprostředkovateli verze 1 • TOPIC-Rozpoznává pouze zástupné znaky na úrovni témat, jak jsou použity ve verzi zprostředkovatele 2	Která verze syntaxe zástupných znaků se má použít.

Notes:

1. Důležité informace o použití vlastnosti sslFipsRequired najdete v části "Omezení adaptéru prostředků IBM MQ" na stránce 427.
2. Informace o tom, jak nakonfigurovat adaptér prostředků tak, aby mohl vyhledat uživatelskou proceduru, naleznete v části "Konfigurace IBM MQ classes for JMS pro použití uživatelských procedur kanálu" na stránce 272.

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 cílového objektu

Aplikační server používá vlastnosti spravovaného cílového objektu k vytvoření objektu JMS Fronta nebo JMS Téma.

Tabulka 67 na stránce 470 obsahuje seznam vlastností, které jsou společné pro objekt fronty a objekt tématu.

Tabulka 67. Vlastnosti společné pro objekt fronty a objekt tématu

Název majetku	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
CCSID	Řetězec	• **1208** • Identifikátor kódované znakové sady podporovaný prostředím JVM (Java Virtual Machine).	Identifikátor kódované znakové sady pro místo určení.
kódování	Řetězec	• **Nativní** • Řetězec se třemi znaky: – První znak určuje reprezentaci binárních celých čísel: - *N* označuje normální kódování. - *R* označuje zpětné kódování. – Druhý znak určuje reprezentaci pakovaných desetinných celých čísel: - *N* označuje normální kódování. - *R* označuje zpětné kódování. – Třetí znak určuje reprezentaci čísel s pohyblivou řádovou čárkou: - *N* označuje standardní kódování IEEE. - *R* označuje reverzní kódování IEEE. - *3* označuje kódování zSeries . NATIVE je ekvivalentní řetězci NNN.	Reprezentace binárních celých čísel, pakovaných desetinných celých čísel a čísel s pohyblivou řádovou čárkou pro místo určení.
Vypršení	Řetězec	• **APP** -Čas vypršení platnosti zprávy je určen producentem zpráv. • UNLIM-Zpráva nikdy nevyprší. • 0-Zpráva nikdy nevyprší. • Kladné celé číslo představující čas vypršení platnosti zprávy v milisekundách.	Čas vypršení platnosti zprávy odeslané na místo určení.
FAILIFQUIESCE	Řetězec	• **ano** • ne	Zda se pokus o přístup k místu určení nezdaří, pokud je správce front ve stavu uvedení do klidového stavu.

Tabulka 67. Vlastnosti společné pro objekt fronty a objekt tématu (pokračování)

Název majetku	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
Styl messageBody	Řetězec	• **Neuvedeno** • JMS • MQ	Můžete nastavit vlastnost **messageBodyStyle** ve frontách a tématech systému JMS : UNSPECIFIED (výchozí) • Při odesílání produkt IBM MQ classes for JMS vygeneruje a zahrne záhlaví MQRFH2 v závislosti na hodnotě WMQ_TARGET_CLIENT. • Při příjmu IBM MQ classes for JMS nastavte vlastnosti zprávy JMS podle hodnot v serveru MQRFH2, pokud existují. MQRFH2 není prezentován jako součást těla zprávy JMS . JMS • Při odesílání produkt IBM MQ classes for JMS automaticky vygeneruje záhlaví MQRFH2 a zahrne je do zprávy IBM MQ . • Při příjmu IBM MQ classes for JMS nastavte vlastnosti zprávy JMS podle hodnot v serveru MQRFH2, pokud existují. MQRFH2 není prezentován jako součást těla zprávy JMS . MQ • Při odesílání IBM MQ classes for JMS negenerujte MQRFH2. • Při příjmu IBM MQ classes for JMS prezentuje MQRFH2 jako součást těla zprávy JMS .

Tabulka 67. Vlastnosti společné pro objekt fronty a objekt tématu (pokračování)

Název majetku	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
trvání, perzistence	Řetězec	• **APP** -Trvání zprávy je určeno producentem zpráv. • QDEF-Trvání zprávy je určeno atributem **DefPersistence** fronty IBM MQ . • PERS-Zpráva je trvalá. • Zpráva typu NON-A je dočasná. • HIGH-Trvání zprávy je určeno atributem **NonPersistentMessageClass** fronty IBM MQ podle vysvětlení v části “JMS trvalých zpráv” na stránce 244.	Perzistence zprávy odeslané do místa určení.
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-Určete, zda jsou povolena asynchronní vložení, odkazem na definici fronty. • TOPIC-Určete, zda jsou povolena asynchronní vložení, odkazem na definici tématu. • DESTINATION-Určete, zda jsou povolena asynchronní vložení, odkazem na definici fronty nebo tématu. • DISABLED-Asynchronní vložení nejsou povolena. • ENABLED-Asynchronní vložení jsou povolena.	Zda mohou producenti zpráv používat asynchronní operace vložení k odesílání zpráv do tohoto místa určení.
READAHEADALLOWED	celé číslo	• **DESTINATION** -Určete, zda je povoleno dopředné čtení, pomocí odkazu na definici fronty nebo tématu. • DISABLED-Dopředné čtení není povoleno. • POVOLENO-Dopředné čtení je povoleno. • QUEUE-Určete, zda je dopředné čtení povoleno odkazem na definici fronty. • TOPIC-Určete, zda je dopředné čtení povoleno s odkazem na definici tématu.	Určuje, zda mohou spotřebitelé zpráv a prohlížeče front používat dopředné čtení k získání dočasných zpráv z místa určení do interní vyrovnávací paměti před jejich přijetím.
receiveCCSID	celé číslo	• **0** -Použít prostředí JVM `Charset.defaultCharset` • 1208- UTF-8 • Podporovaný identifikátor kódované znakové sady	Cílová vlastnost, která nastavuje cílový CCSID pro převod zpráv správce front. Hodnota je ignorována, pokud není parametr **receiveConversion** nastaven na hodnotu `QMGR`.

Tabulka 67. Vlastnosti společné pro objekt fronty a objekt tématu (pokračování)			
Název majetku	**Typ**	**Platné hodnoty (výchozí hodnota tučně)**	**Popis**
receiveConversion	Řetězec	• **CLIENT_MSG** • QMGR	Cílová vlastnost, která určuje, zda má správce front provést převod dat.
targetClient	Řetězec	• **JMS** -Cílem zprávy je aplikace JMS . • MQ -Cílem zprávy je jiná aplikace nežJMS IBM MQ .	Určuje, zda je cílem zprávy odeslané do místa určení aplikace JMS . Zpráva s cílem, který je aplikací JMS , obsahuje záhlaví MQRFH2 .

V tabulce Tabulka 68 na stránce 473 jsou uvedeny vlastnosti specifické pro objekt fronty.

Tabulka 68. Vlastnosti specifické pro objekt fronty			
Název majetku	**Typ**	**Platné hodnoty (výchozí hodnota tučně)**	**Popis**
baseQueueManagerName	Řetězec	• **"" (prázdný řetězec)** • Název správce front	Název správce front, který vlastní základní frontu IBM MQ .
Název baseQueue	Řetězec	• **"" (prázdný řetězec)** • Název fronty	Název základní fronty IBM MQ .

Tabulka 69 na stránce 473 uvádí vlastnosti, které jsou specifické pro objekt Topic.

Tabulka 69. Vlastnosti specifické pro objekt Topic			
Název majetku	**Typ**	**Platné hodnoty (výchozí hodnota tučně)**	**Popis**
Název baseTopic	Řetězec	• **"" (prázdný řetězec)** • Název tématu	Název základního tématu.
brokerCCDurSubQueue >	Řetězec	• **SYSTEM.JMS.D.CC.SUBSCRIBER.QUEUE** • Název fronty	Název fronty, ze které spotřebitel připojení přijímá zprávy trvalého odběru.
brokerDurSubQueue	Řetězec	• **SYSTEM.JMS.D.SUBSCRIBER.QUEUE** • Název fronty	Název fronty, ze které trvalý odběratel tématu přijímá zprávy. Další informace viz vlastnost **BROKEDURRSUBQ** v dokumentaci k produktu IBM MQ Explorer .
Fronta brokerPub	Řetězec	• **Nenastaveno** • Název fronty	Název fronty, kam 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 .

Tabulka 69. Vlastnosti specifické pro objekt Topic (pokračování)			
Název majetku	**Typ**	**Platné hodnoty (výchozí hodnota tučně)**	**Popis**
brokerPubQueueManager	Řetězec	• **"" (prázdný řetězec)** • Název správce front	Název správce front, který vlastní frontu, do které jsou odesílány zprávy publikované v tématu.
brokerVersion	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 .

Následující příklad ukazuje sadu vlastností objektu fronty:

```
expiry:               UNLIM
persistence:          QDEF
baseQueueManagerName: ExampleQM
baseQueueName:        SYSTEM.JMS.TEMPQ.MODEL
```

Následující příklad zobrazuje sadu vlastností objektu Téma:

```
expiry:               UNLIM
persistence:          NON
baseTopicName:        myTestTopic
```

Související úlohy

Určení, že se za běhu v klientu MQI používají pouze specifikace CipherSpecs s certifikací FIPS.

Konfigurace prostředků JMS v produktu WebSphere Application Server

Související odkazy

Standard FIPS (Federal Information Processing Standards) pro AIX, Linux, and Windows

Konfigurace vlastnosti targetClientOdpovídající vlastnosti pro specifikaci aktivace

Vlastnost **targetClientMatching** můžete nakonfigurovat pro specifikaci aktivace tak, aby záhlaví MQRFH2 bylo zahrnuto do zpráv odpovědi, pokud zprávy požadavku neobsahují záhlaví MQRFH2 . To znamená, že všechny vlastnosti zprávy, které aplikace definuje ve zprávě odpovědi, 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 s odpovědí do cíle JMS vytvořeného z pole JMSReplyTo zprávy požadavku, musí zprávy s odpovědí obsahovat záhlaví MQRFH2 , a to i v případě, že zprávy s požadavkem nikoli, jinak dojde ke ztrátě vlastností zpráv, které aplikace definovala ve zprávě s odpovědí.

Vlastnost **targetClientMatching** definuje, zda má zpráva odpovědi odeslaná do fronty určené polem záhlaví JMSReplyTo příchozí zprávy záhlaví MQRFH2 pouze v případě, že příchozí zpráva má záhlaví MQRFH2 . Tuto vlastnost můžete konfigurovat pro specifikaci aktivace v produktu WebSphere Application Server traditional i v produktu WebSphere Liberty.

Nastavíte-li hodnotu vlastnosti **targetClientMatching** na `false`, záhlaví MQRFH2 lze zahrnout do zprávy odpovědi odeslané do cíle JMS vytvořeného ze záhlaví JMSReplyTo příchozí zprávy požadavku, která neobsahuje MQRFH2. Důvodem je, že vlastnost **targetClient** v cíli JMS je nastavena na hodnotu `0`, což znamená, že zprávy obsahují záhlaví MQRFH2 . Přítomnost záhlaví MQRFH2 v odchozí zprávě umožňuje při odeslání do fronty IBM MQ ukládání vlastností zpráv definovaných uživatelem.

Je-li vlastnost **targetClientMatching** nastavena na hodnotu `true` a zpráva požadavku neobsahuje záhlaví MQRFH2 , nebude záhlaví MQRFH2 zahrnuto do zprávy odpovědi.

Procedura

- V produktu WebSphere Application Server traditionalpomocí konzoly pro správu definujte vlastnost **targetClientMatching** jako přizpůsobenou vlastnost ve specifikaci aktivace IBM MQ :
  a) V navigačním podokně klepněte na volbu **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 souboru WebSphere Libertyzadejte vlastnost **targetClientMatching** pro 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 213
Namísto načítání míst určení jako spravovaných objektů z oboru názvů rozhraní JNDI (Java Naming and Directory Interface) může aplikace JMS používat relaci k dynamickému vytváření míst určení za běhu. Aplikace může pomocí identifikátoru URI (Uniform Resource Identifier) identifikovat frontu IBM MQ nebo téma a volitelně zadat jednu či více vlastností objektu Queue nebo Topic.

“Konfigurace adaptéru prostředků pro odchozí komunikaci” na stránce 457
Chcete-li konfigurovat odchozí komunikaci, definujte vlastnosti objektu ConnectionFactory a spravovaného cílového objektu.

IBM MQ pozastavení objektů typu message-driven bean v WebSphere Liberty

Vlastnost **maxSequentialDeliveryFailures** pro specifikaci aktivace definuje maximální počet postupných selhání doručení zpráv do instance objektu typu message-driven bean (MDB), které adaptér prostředků toleruje před pozastavením objektu MDB.

Než začnete

Musíte mít na paměti sadu událostí, které mohou způsobit pozastavení MDB v produktu WebSphere Liberty. Adaptér prostředků považuje za selhání doručení zprávy jednu z následujících možností:

- Z metody **onMessage** objektu MDB byla vyvolána nekontrolovaná výjimka.
- `JMSException` vyskytující se ve zpracování adaptéru prostředků před doručením zprávy do objektu MDB.
- `JMSException` vyskytující se ve zpracování adaptéru prostředků po doručení zprávy do objektu MDB.
- Transakce XA nebo lokální transakce používaná k přijetí odvolané zprávy.
- Na aplikačním serveru není k dispozici žádný podproces pro doručení zprávy do objektu MDB.

Informace o této úloze

Výchozí hodnota vlastnosti **maxSequentialDeliveryFailures** je *-1*, což znamená, že objekt MDB není nikdy pozastaven. S jakoukoli jinou zápornou hodnotou se zachází stejně jako s hodnotou *-1*. Hodnota:

- Hodnota *0* znamená, že se objekt MDB pozastaví při první chybě.
- *1* znamená, že se objekt typu message-driven bean pozastaví na dvou po sobě následujících chybách.
- *2* znamená, že se objekt MDB pozastaví na třech po sobě následujících chybách atd.

Tuto vlastnost můžete konfigurovat pro specifikaci aktivace pouze v produktu WebSphere Libertya v případě, že úroveň parametru Liberty je 18.0.0.4nebo vyšší.

Upozornění: Nastavíte-li atribut na jinou než výchozí hodnotu v jiném prostředí aplikačního serveru než Liberty, hodnota bude ignorována a do protokolu bude zapsána varovná zpráva.

Kromě toho je možné nainstalovat adaptér prostředků IBM MQ do adresáře WebSphere Liberty jako generický adaptér prostředků. Tímto zakážete všechny schopnosti integrace IBM MQ a WebSphere Application Server a zabráníte adaptéru prostředků, aby byl schopen zjistit, že je spuštěn v produktu Liberty. Proto nastavení **maxSequentialDeliveryFailures** na hodnotu větší nebo rovnou *0* není podporováno a výsledkem je varovná zpráva v protokolu.

Procedura

- V souboru WebSphere Libertyzadejte vlastnost **maxSequentialDeliveryFailures** pro definici specifikace aktivace v rámci `server.xml`.

  Příklad:

```
<jmsActivationSpec>
     <properties.wmqJms destinationRef="jndi/MDBQ"
                        transportType="BINDINGS"
                        queueManager="MQ21"
                        maxSequentialDeliveryFailures="1"/>
</jmsActivationSpec>
```

Související pojmy

“Konfigurace adaptéru prostředků pro odchozí komunikaci” na stránce 457
Chcete-li konfigurovat odchozí komunikaci, definujte vlastnosti objektu ConnectionFactory a spravovaného cílového objektu.

Ověření instalace adaptéru prostředků

Program testu ověření instalace (IVT) pro adaptér prostředků IBM MQ je dodáván jako soubor EAR. Chcete-li použít program, musíte jej implementovat a definovat některé objekty jako prostředky JCA .

Informace o této úloze

Program ověřovacího testu instalace (IVT) je dodáván jako soubor podnikového archivu (EAR) s názvem `wmq.jakarta.jmsra.ivt.ear` (Jakarta Messaging 3.0) nebo `wmq.jmsra.ivt.ear` (JMS 2.0). Tento soubor je nainstalován s produktem IBM MQ classes for JMS ve stejném adresáři jako soubor RAR adaptéru prostředků IBM MQ , `wmq.jakarta.jmsra.rar` (Jakarta Messaging 3.0) nebo `wmq.jmsra.rar` (JMS 2.0). Informace o tom, kde jsou tyto soubory nainstalovány, viz “Instalace adaptéru prostředků IBM MQ” na stránce 431.

Musíte implementovat program IVT na svém aplikačním serveru. Program IVT zahrnuje servlet a objekt MDB, který testuje, zda lze zprávu odeslat do fronty IBM MQ a přijmout z ní. Pomocí programu IVT můžete ověřit, zda byl adaptér prostředků IBM MQ správně nakonfigurován pro podporu distribuovaných transakcí. Pokud implementujete adaptér prostředků IBM MQ na jiný aplikační server nežIBM , může vás služba IBM požádat o předvedení IVT, které pracuje na ověření, že je váš aplikační server správně nakonfigurován.

Než budete moci spustit program IVT, musíte definovat objekt ConnectionFactory , objekt Queue a případně objekt Activation Specification jako prostředky JCA a zajistit, aby váš aplikační server vytvořil objekty JMS z těchto definic a vázal je do oboru názvů JNDI . Můžete zvolit vlastnosti objektů tak, aby odpovídaly nastavení hostitele a portu vašeho vlastního správce QueueManager, ale následující sada 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
```

Mechanizmus používaný k definování objektů ConnectionFactory, Queue a Activation Specification se liší v závislosti na aplikačním serveru. Chcete-li například nastavit tyto vlastnosti v rámci souboru WebSphere Liberty, přidejte do souboru `server.xml` aplikačního serveru následující položky:

JM 3.0
```
<!-- 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.jakarta.jmsra.ivt/WMQ_IVT_MDB/WMQ_IVT_MDB">
   <properties.wmqJms destinationRef="IVTQueue"
transportType="CLIENT"
queueManager="QM1"
hostName="localhost"
port="1550"
maxPoolDepth="1"/>
</jmsActivationSpec>
```

JMS 2.0
```
<!-- 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>
```

Při výchozím nastavení program IVT očekává, že objekt ConnectionFactory bude svázán v oboru názvů JNDI s názvem jms/ivt/IVTCF a objektem Queue, který má být svázán s názvem jms/ivt/IVTQueue. Můžete použít různé názvy, ale pokud tak učiníte, musíte zadat názvy objektů na počáteční stránce programu IVT a odpovídajícím způsobem upravit soubor EAR.

Po implementaci programu IVT a vytvoření objektů JMS aplikačním serverem a jejich svázání do oboru názvů JNDI můžete spustit program IVT provedením následujících kroků.

Postup

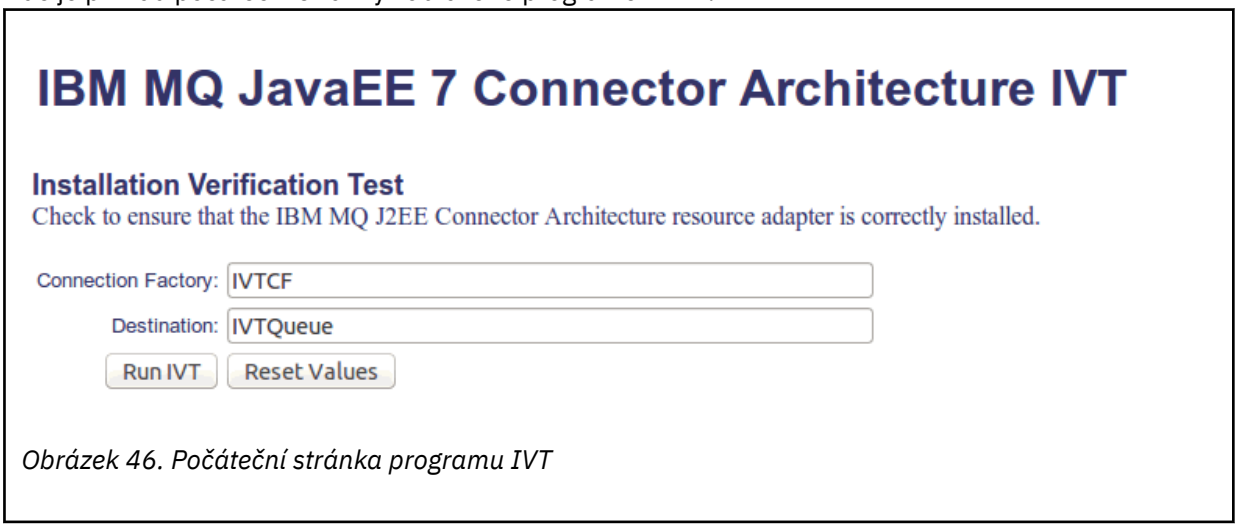
1. Spusťte program IVT zadáním URL do webového prohlížeče v následujícím formátu:

   ```
   http://app_server_host: port/WMQ_IVT/
   ```

   kde *název_aplikačního_serveru* je adresa IP nebo název hostitele systému, na kterém je spuštěn váš aplikační server, a *port* je číslo portu TCP, na kterém aplikační server naslouchá. Příklad:

   ```
   http://localhost:9080/WMQ_IVT/
   ```

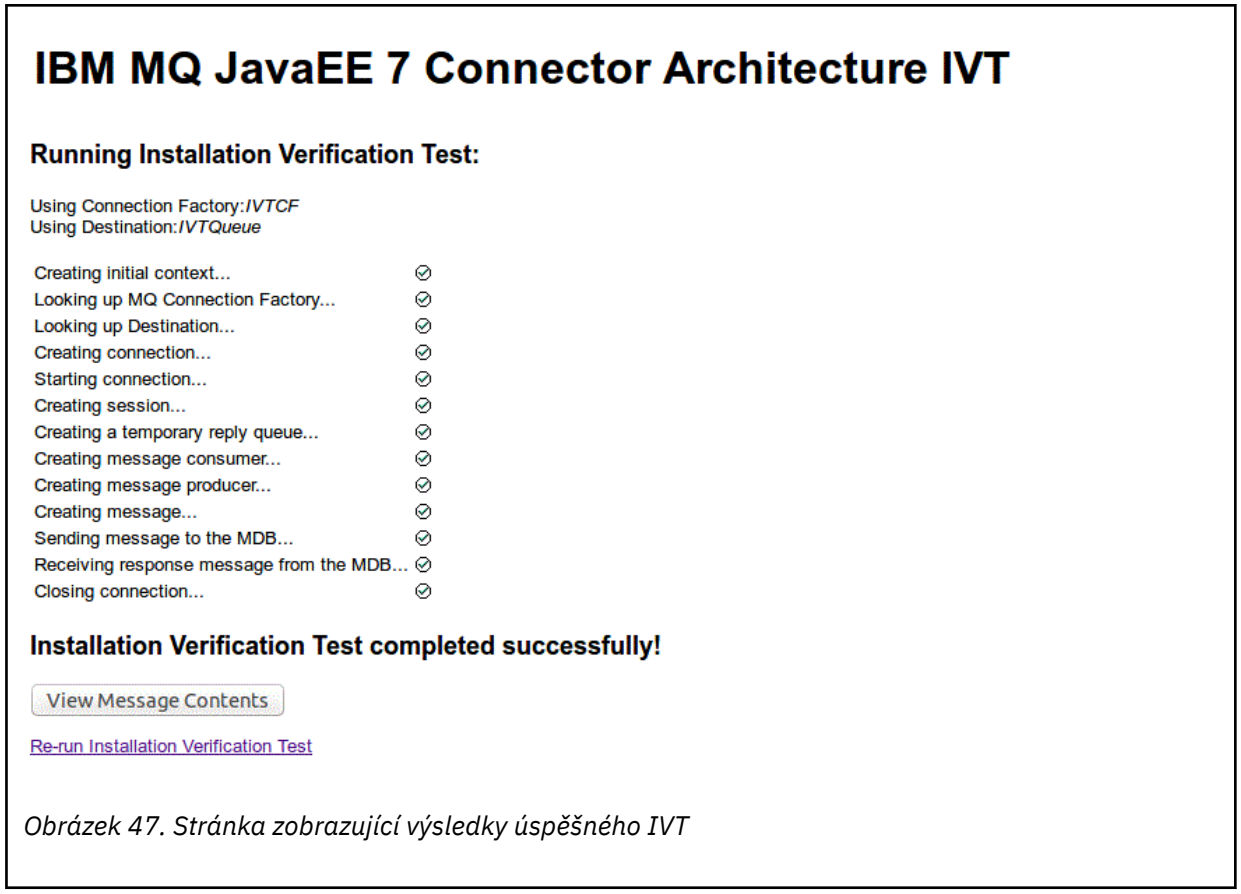
   Zde je příklad počáteční stránky zobrazené programem IVT.

   Obrázek 46. Počáteční stránka programu IVT

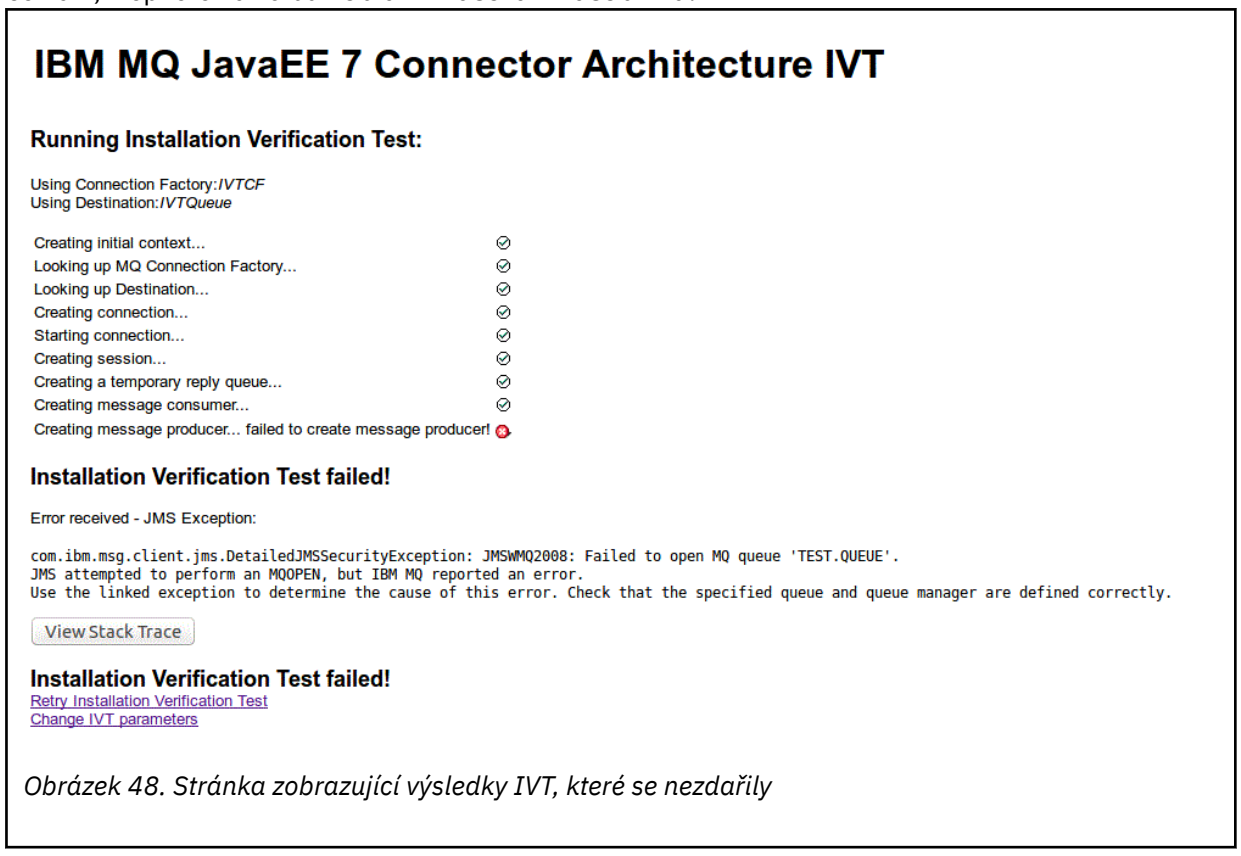
2. Chcete-li spustit test, klepněte na volbu **Spustit IVT**.

   Zde je příklad stránky, která se zobrazí, pokud je IVT úspěšný.



Obrázek 47. Stránka zobrazující výsledky úspěšného IVT

Zde je příklad stránky, která se zobrazí, pokud se IVT nezdaří. Chcete-li získat další informace o příčině selhání, klepněte na volbu **Zobrazit trasování zásobníku**.



Obrázek 48. Stránka zobrazující výsledky IVT, které se nezdařily

Windows Instalace a testování adaptéru prostředků na serveru GlassFish

Chcete-li nainstalovat adaptér prostředků IBM MQ na serveru 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 ověření instalace (IVT).

Než začnete

- Tyto pokyny jsou určeny pro server GlassFish Server verze 4.
- Tato verze serveru GlassFish nepodporuje Jakarta EE.

Informace o této úloze

Tato úloha předpokládá, že máte spuštěný aplikační server GlassFish Server a že jste pro něj obeznámeni se standardními úlohami administrace. Tato úloha také předpokládá, že máte instalaci produktu IBM MQ na lokálním systému a že jste obeznámeni se standardními úlohami administrace.

Poznámka: Chcete-li provést následující kroky úlohy, musíte mít funkční instalaci produktu IBM MQ s nakonfigurovanými následujícími objekty:

- Správce front s názvem QM spuštěný na portu 1414, který používá kanál SYSTEM.DEF.SVRCONN, který se připojuje pomocí přenosu klienta.
- Fronta s názvem Q1.

Postup

1. Spusťte program shellu GlassFish Server **asadmin** .
   a) Otevřete příkazový řádek Windows a přejděte do adresáře *GlassFish*/bin , kde *GlassFish* je adresář, kde je nainstalován server GlassFish Server verze 4.
   b) Zadejte příkaz **asadmin** do příkazového řádku.
   Příkaz **asadmin** otevře v příkazovém řádku program shellu, který vám umožní vytvořit novou doménu.
   Ve vašem systému je spuštěn server GlassFish Server verze 4.
2. Vytvořte a poté spusťte doménu.
   a) K vytvoření nové domény použijte příkaz **create-domain** se zadaným portem a názvem domény. Na příkazový řádek zadejte následující příkaz:

   ```
   create-domain --adminport port domain_name
   ```

   kde *port* je číslo portu a *název_domény* je název, který má doména používat.

   Poznámka: Příkaz **create-domain** má k sobě přidruženo mnoho volitelných parametrů. Pro tuto úlohu však potřebujete pouze parametr `-- adminport` . Další informace naleznete v dokumentaci k produktu GlassFish Server verze 4.

   Pokud je port, který jste uvedli, používán, zobrazí se následující zpráva:

   `Port pro název_domény port se používá`

   Pokud je zadaný název domény používán, zobrazí se zpráva s informací, že zadaný název je již používán, a také seznam všech názvů domén, které jsou v současné době nedostupné.
   b) Když jste vyzváni k zadání jména uživatele a hesla, zadejte pověření, která se mají použít pro přihlášení k aplikačnímu serveru prostřednictvím webového prohlížeče.
   Pokud je příkaz úspěšně dokončen, na příkazovém řádku se zobrazí zpráva shrnující vytvoření domény, včetně zprávy `Command create-domain executed successfully..`
   Úspěšně jste vytvořili doménu.
   c) Spusťte doménu zadáním následujícího příkazu do příkazového řádku:

```
start-domain domain_name
```

kde *domain_name* je název domény, který jste uvedli dříve.

3. Pro přístup k aplikačnímu serveru GlassFish použijte webový prohlížeč.
   a) V adresním řádku webového prohlížeče zadejte následující příkaz:

   ```
   localhost:port
   ```

   kde *port* je port, který jste uvedli dříve při vytváření domény.

   Zobrazí se konzola GlassFish .

   b) Po načtení konzoly GlassFish a po zobrazení výzvy 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) Klepnutím na tlačítko **Implementovat** otevřete 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** , abyste otevřeli 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-Pool do pole **Název fondu** .

   e) V poli **Adaptér prostředků** vyberte volbu `wmq.jmsra`.

   f) Do pole **Definice připojení** zadejte `javax.jms.ConnectionFactory`.

   g) Vyberte volbu **Další**a poté vyberte volbu **Dokončit**.
6. Vytvořte prostředky konektoru.
   a) Na panelu nástrojů v nabídce **Konektory** vyberte volbu **Prostředek konektoru** a otevřete stránku **Prostředky konektoru** .

   b) Výběrem volby **Nový**otevřete 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, jak ukazuje následující příklad:

   - 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 pro vlastní nastavení konfigurace používáte správné hodnoty, které se mohou lišit od těch, které jsou uvedeny v tomto příkladu.

   g) Na panelu nástrojů v části **Konektory**vyberte položku nabídky **Prostředky objektu administrátora** a otevřete stránku **Prostředky objektu administrátora** .

h) Na stránce **Prostředky objektu administrátora** klepněte na tlačítko **Nový** a otevřete stránku **Nový prostředek objektu administrátora** .

i) Do pole **Název rozhraní JNDI** zadejte `IVTQueue`.

j) Do pole **Adaptér prostředků** zadejte hodnotu `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, jak ukazuje následující příklad:

- název: název; hodnota: IVTQueue
- název: baseQueueManagerName; hodnota QM
- název: baseQueueNázev; hodnota: Q1

Poznámka: Ujistěte se, že pro vlastní nastavení konfigurace používáte správné hodnoty, které se mohou lišit od těch, které jsou uvedeny v tomto příkladu.

n) Klepněte na tlačítko **OK**.

o) Zaškrtněte políčko **Povoleno** a poté klepněte na volbu **Povolit**.

7. Nasaďte soubor EAR `wmq.jmsra.ivt.ear` na server GlassFish .

a) Klepnutím na volbu **Aplikace** na panelu nástrojů zobrazíte stránku **Aplikace** .

b) Klepnutím na volbu **Implementovat** přidejte aplikaci IVT.

c) V poli **Umístění** přejděte na položku `wmq.jmsra.ivt.ear`a vyberte ji.

d) V poli **Virtual Servers** vyberte **server**a pak klepněte na tlačítko **OK**.

8. Spusťte program IVT.

a) Klepnutím na volbu **Aplikace** na panelu nástrojů zobrazíte stránku **Aplikace** .

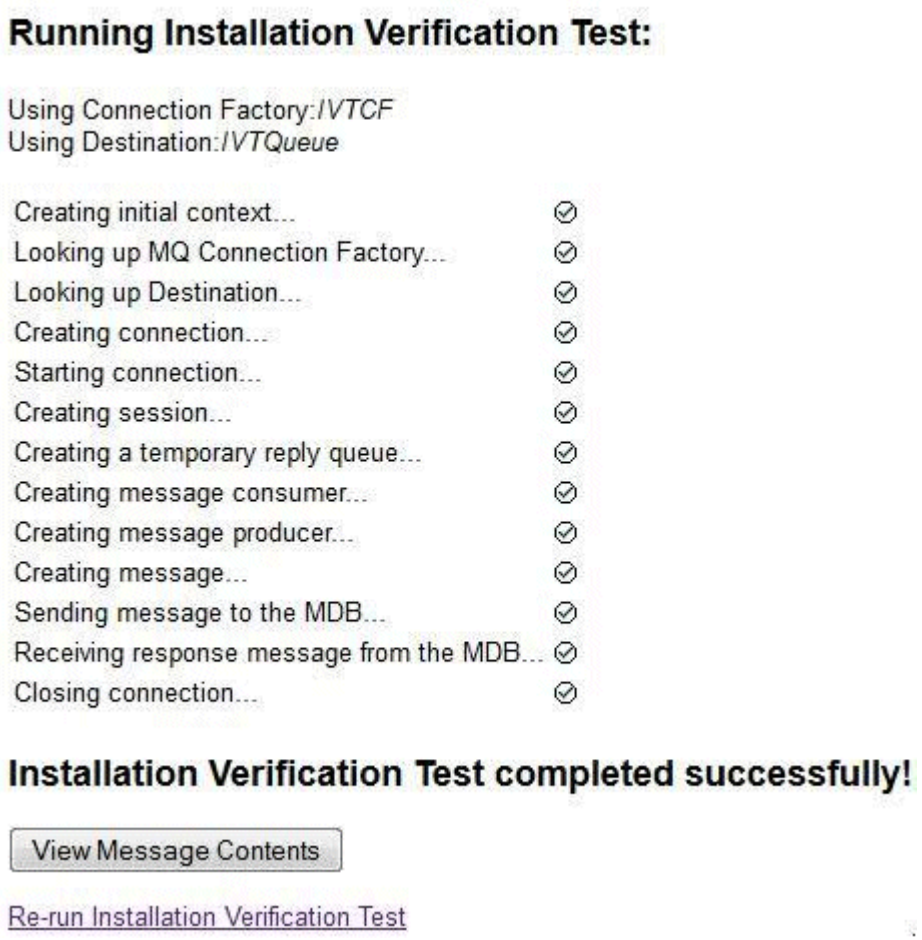
b) Klepněte na volbu `wmq.jmsra.ivt` v tabulce Implementované aplikace.

c) Klepněte na tlačítko **Spustit** v tabulce Moduly a komponenty.

d) Vyberte odkaz http: link.

e) Klepněte na volbu **Spustit IVT**.

Spustili jste program IVT a pokud jste úspěšní, zobrazí se následující výstup:



Obrázek 49. Úspěšný výstup IVT

Instalace a testování adaptéru prostředků v modulu WildFly

Pokud instalujete adaptér prostředků IBM MQ v WildFly V10, musíte nejprve provést některé změny v konfiguračním souboru, abyste přidali definici subsystému pro adaptér prostředků IBM MQ . Poté můžete implementovat adaptér prostředků a otestovat jej instalací a spuštěním aplikace IVT (installation verification test).

Než začnete

- Tyto pokyny jsou určeny pro WildFly V10.
- Tato verze WildFly nepodporuje Jakarta EE.

Informace o této úloze

Tato úloha předpokládá, že máte spuštěný aplikační server WildFly a že jste pro něj obeznámeni se standardními úlohami administrace. Tato úloha také předpokládá, že máte instalaci produktu IBM MQ a že jste obeznámeni se standardními úlohami administrace.

Postup

1. Vytvořte správce front IBM MQ s názvem ExampleQMa nastavte jej podle popisu v části “Konfigurace správce front pro přijetí připojení klienta na platformě Multiplatforms” na stránce 1034.

   Při nastavování správce front si všimněte následujících bodů:

   - Modul listener musí být spuštěn na portu 1414.

- Kanál, který se má použít, se nazývá SYSTEM.DEF.SVRCONN.
- Fronta používaná aplikací IVT má název TEST.QUEUE.

Fronta modelu SYSTEM.DEFAULT.MODEL.QUEUE také musí být uděleno oprávnění DSP a PUT, aby mohla tato aplikace 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. Implementujte adaptér prostředků na server zkopírováním souboru wmq.jmsra.rar do adresáře *WildFly_Home*/standalone/deployments.
4. Implementujte aplikaci IVT tak, že zkopírování souboru wmq.jmsra.ivt.ear do adresáře *WildFly_Home*/standalone/deployments.

5. Spusťte aplikační server vyvoláním příkazového řádku, přejděte do adresáře *WildFly_Home*/bin a spusťte příkaz:

```
standalone.bat -c standalone-full.xml
```

6. Spusťte aplikaci IVT.

   Další informace viz téma “Ověření instalace adaptéru prostředků” na stránce 476. Pro WildFlyje výchozí URL `http://localhost:8080/WMQ_IVT/`.

Společné použití IBM MQ a WebSphere Application Server

Prostřednictvím poskytovatele systému zpráv IBM MQ v produktu WebSphere Application Servermohou aplikace systému zpráv Java Message Service (JMS) používat systém IBM MQ jako externího poskytovatele prostředků systému zpráv JMS .

Informace o této úloze

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

Výhodou použití správce front IBM MQ je, že připojení aplikací JMS se může plně podílet na funkčnosti sítě IBM MQ , což umožňuje aplikacím vyměňovat si zprávy se správci front, kteří jsou spuštěni na mnoha platformách.

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

Standardně zprávy systému JMS , které jsou zadrženy ve frontách systému IBM MQ , používají záhlaví MQRFH2 k uchování některých informací o záhlaví zpráv produktu JMS . Mnoho starších aplikací IBM MQ nemůže zpracovat zprávy s těmito záhlavími a vyžaduje jejich vlastní charakteristická záhlaví, například MQCIH for CICS Bridge nebo MQWIH for IBM MQ Workflow. Další informace o těchto zvláštních aspektech naleznete v tématu Mapování JMS zpráv na IBM MQ zprávy.

Související úlohy

Konfigurace prostředků JMS v adresáři 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 s výchozím poskytovatelem systému zpráv, který je součástí produktu WebSphere Application Server, nebo jako alternativu k tomuto poskytovateli systému zpráv.

Poskytovatel systému zpráv IBM MQ je instalován jako součást produktu WebSphere Application Server. To zahrnuje verzi adaptéru prostředků IBM MQ a funkci IBM MQ Extended Transactional Client, která umožňuje správci front účastnit se transakcí XA spravovaných aplikačním serverem. Pomocí adaptéru prostředků lze objekty typu message-driven bean konfigurovat tak, aby používaly specifikace aktivace nebo porty modulu listener.

Aby byl aplikační server podporován, je třeba na aplikační server implementovat a úspěšně spustit testovací program pro ověření instalace adaptéru prostředků IBM MQ . Po úspěšném spuštění programu ověření instalace adaptéru prostředků IBM MQ se může adaptér prostředků IBM MQ připojit k libovolnému podporovanému správci front IBM MQ .

JMS připojení z WebSphere Application Server do IBM MQ

Před zvážením úrovní produktu IBM MQ , které lze použít s produktem WebSphere Application Server, je důležité pochopit, jak se mohou aplikace Java Message Service (JMS) spuštěné v rámci aplikačního serveru připojovat ke správcům front produktu IBM MQ .

Aplikace systému JMS , které potřebují přístup k prostředkům správce front systému IBM MQ , tak mohou učinit pomocí jednoho z následujících typů přenosu:

Vazby

Tento přenos lze použít, pokud jsou aplikační server a správce front nainstalovány na stejném počítači a obrazu operačního systému. Při použití režimu BINDINGS se veškerá komunikace mezi těmito dvěma produkty provádí pomocí IPC (Inter-Process Communication).

Poskytovatel systému zpráv IBM MQ nezahrnuje nativní knihovny nezbytné pro připojení ke správci front IBM MQ v režimu BINDINGS. Chcete-li používat připojení v režimu BINDINGS, musí být produkt IBM MQ nainstalován na stejném počítači jako aplikační server a cesta k nativní knihovně adaptéru prostředků musí být nakonfigurována tak, aby ukazovala na adresář IBM MQ , ve kterém jsou tyto knihovny umístěny. Další informace naleznete v dokumentaci k produktu WebSphere Application Server :

- Informace o systému WebSphere Application Server traditionalnaleznete v tématu Konfigurace poskytovatele systému zpráv IBM MQ s nativními knihovnami.
- Informace o produktu WebSphere Libertynaleznete v tématu Implementace aplikací JMS na server Liberty pro použití IBM MQ poskytovatele systému zpráv.

z/OS Chcete-li v systému z/OSpřipojit továrnu připojení systému WebSphere Application Server ke správci front systému IBM MQ v režimu vazeb, je třeba určit správné knihovny IBM MQ ve zřetězení knihovny WebSphere Application Server STEPLIB. Další informace viz knihovny IBM MQ a soubor WebSphere Application Server pro z/OS STEPLIB v dokumentaci k produktu WebSphere Application Server .

CLIENT

Přenos klienta používá protokol TCP/IP ke komunikaci mezi WebSphere Application Server a IBM MQ. Kromě použití v případě, že jsou aplikační server a správce front umístěny v různých počítačích, lze režim CLIENT použít i v případě, že jsou oba produkty nainstalovány ve stejném počítači a obrazu operačního systému.

Aplikace systému JMS mohou také určit typ transportu BINDINGS_THEN_CLIENT. Při použití tohoto typu transportu se aplikace 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 Která verze adaptéru prostředků WebSphere MQ (RA) je dodávána s produktem WebSphere Application Server?.

Pomocí následujících příkazů Jython a JACL můžete určit úroveň adaptéru prostředků, který produkt WebSphere Application Server aktuálně používá:

Jython

```
wmqInfoMBeansUnsplit = AdminControl.queryNames("WebSphere:type=WMQInfo,*")
wmqInfoMBeansSplit = AdminUtilities.convertToList(wmqInfoMBeansUnsplit)
for wmqInfoMBean in wmqInfoMBeansSplit: print wmqInfoMBean; print
AdminControl.invoke(wmqInfoMBean, 'getInfo', '')
```

Poznámka: Po zadání tohoto příkazu musíte dvakrát klepnout na tlačítko **Vrátit** , abyste jej spustili.

Seznam JACL

```
set wmqInfoMBeans [$AdminControl queryNames WebSphere:type=WMQInfo,*]
foreach wmqInfoMBean $wmqInfoMBeans {
puts $wmqInfoMBean;
puts [$AdminControl invoke $wmqInfoMBean getInfo [] []]
}
```

Aktualizace adaptéru prostředků

Aktualizace adaptéru prostředků IBM MQ , který je nainstalován s aplikačním serverem, jsou zahrnuty v opravných sadách WebSphere Application Server Fix Pack. Aktualizace adaptéru prostředků IBM MQ pomocí volby **Aktualizovat adaptér prostředků ...** v administrativní konzole WebSphere Application Server se nedoporučuje, protože to znamená, že aktualizace poskytované v produktu WebSphere Application Server Fix Pack nebudou mít žádný účinek.

Proměnná MQ_INSTALL_ROOT

V produktu WebSphere Application Server 7.0se parametr MQ_INSTALL_ROOT používá pouze k vyhledání nativních knihoven a je přepsán libovolnou cestou k nativní knihovně konfigurovanou v adaptéru prostředků.

Připojení z WebSphere Application Server do IBM MQ

Upozornění:

1. Jakákoli podporovaná verze produktu WebSphere Application Server může použít adaptér prostředků IBM MQ , který je s ním dodáván v balíku, pro připojení k jakékoli podporované verzi produktu IBM MQ.
2. Je-li použit režim vazeb, musí určité knihovny v produktu WebSphere Application Server odpovídat verzi správce front, ke kterému se připojuje:
   - Produkt WebSphere Application Server musí být nakonfigurován tak, aby načetl nativní knihovny dodávané s produktem IBM MQ 9.3. Další informace viz “Konfigurace knihoven JNI (Java Native Interface)” na stránce 92.
   - z/OS V systému z/OSmusíte zadat správné knihovny IBM MQ ve zřetězení WebSphere Application Server STEPLIB.

     Podrobné informace o knihovnách systému IBM MQ , které potřebujete, naleznete v části IBM MQ a WebSphere Application Server for z/OS STEPLIB .

     Máte-li knihovny pro jednu verzi produktu IBM MQ v LINKLIST (LINKLST), můžete se připojit k jiné verzi produktu IBM MQ přepsáním knihoven pomocí STEPLIB.
3. Verze adaptéru prostředků IBM MQ je nezávislá na verzích nativní (sdílené) knihovny poskytnutých instalací správce front.

   Například WebSphere Application Server 8.5s adaptérem prostředků IBM MQ 8.0 může i nadále spravovat připojení vazeb ke správci front IBM MQ 9.0 pomocí nativních knihoven IBM MQ 9.0 .

Další informace viz téma “Příkaz podpory adaptéru prostředků IBM MQ” na stránce 425.

Typy transportu BINDINGS a CLIENT lze použít pro připojení k produktu IBM MQ z libovolné verze produktu WebSphere Application Server. Pro typ transportu BINDINGS platí následující omezení:

- Produkt IBM MQ musí být nainstalován na stejném počítači jako aplikační server.
- Produkt WebSphere Application Server musí být nakonfigurován tak, aby načetl nativní knihovny dodávané s produktem IBM MQ.
- z/OS Chcete-li v systému z/OSpřipojit továrnu připojení produktu WebSphere Application Server ke správci front IBM MQ v režimu vazeb, musí být ve zřetězení knihovny WebSphere Application Server STEPLIB zadány správné knihovny IBM MQ .

V následující tabulce jsou uvedeny verze produktu WebSphere Application Server , v nichž je podporováno spuštění jednotlivých verzí adaptéru prostředků IBM MQ .

Tabulka 70. Mapování verzí WebSphere Application Server na verze adaptéru prostředků IBM MQ .	
Verze adaptéru prostředků IBM MQ	**Ve které verzi produktu WebSphere Application Server může být tato verze adaptéru prostředků spuštěna?**
IBM MQ 9.0 a novější	Adaptér prostředků může být spuštěn v: • Libovolná Java EE 7 vyhovující verze produktu WebSphere Liberty. • WebSphere Application Server traditional 9.0
IBM MQ 8.0	Adaptér prostředků může být spuštěn v libovolné Java EE 7 vyhovující verzi produktu WebSphere Liberty . Adaptér prostředků IBM MQ 8.0 není podporován pro spuštění v adresáři WebSphere Application Server traditional. Adaptér prostředků, který je již nainstalován v adresáři WebSphere Application Server traditional , by měl být použit pro připojení ke správcům front produktu IBM MQ 8.0 .

Související pojmy

"Příkaz podpory adaptéru prostředků IBM MQ" na stránce 425
Produkt Adaptér prostředků IBM MQ , který musíte použít pro komunikaci mezi aplikací a správcem front, závisí na tom, zda používáte rozhraní API produktu Jakarta Messaging 3.0 nebo JMS 2.0 API.

Související informace

Systémové požadavky pro IBM MQ

Určení počtu připojení TCP/IP vytvořených z WebSphere Application Server na IBM MQ

Pomocí funkce sdílení konverzací může více konverzací sdílet instance kanálu MQI, což je také známé jako připojení TCP/IP.

Informace o této úloze

Aplikace spuštěné uvnitř systémů WebSphere Application Server 7 a WebSphere Application Server 8, které používají normální režim poskytovatele systému zpráv IBM MQ , budou tuto funkci automaticky používat. To znamená, že více aplikací spuštěných v rámci stejné instance aplikačního serveru, které se připojují ke stejnému správci front IBM MQ , může sdílet stejnou instanci kanálu.

Počet konverzací, které lze sdílet v rámci jedné instance kanálu, je určen IBM MQ vlastností kanálu **SHARECNV**. Výchozí hodnota této vlastnosti pro kanály připojení serveru je 10.

Podíváme-li se na počet konverzací vytvořených pomocí WebSphere Application Server 7 a WebSphere Application Server 8, můžete určit počet instancí kanálu, které byly vytvořeny.

Další informace o režimu poskytovatele systému zpráv IBM MQ naleznete v části PROVIDERVERSION normal mode.

Související pojmy

Použití sdílení konverzací
V prostředí, kde je povoleno sdílení konverzací, mohou konverzace sdílet instanci kanálu MQI.

"Sdílení připojení TCP/IP v produktu IBM MQ classes for JMS" na stránce 306
Pro sdílení jednoho připojení TCP/IP lze vytvořit více instancí kanálu MQI.

Továrny připojení JMS

Aplikace spuštěné uvnitř produktu WebSphere Application Server, které k vytváření připojení a relací používají továrnu připojení poskytovatele systému zpráv IBM MQ , mají aktivní konverzace pro každé připojení produktu JMS vytvořené z továrny připojení a pro každou relaci produktu JMS vytvořenou z připojení produktu JMS .

Jedna konverzace pro každé připojení JMS , které bylo vytvořeno z továrny připojení

Každá továrna připojení JMS má přidružený fond připojení, rozdělený do dvou sekcí, volný fond a aktivní fond. Oba fondy jsou na počátku prázdné.

Když aplikace vytvoří připojení JMS z továrny připojení, produkt WebSphere Application Server zkontroluje, zda existuje připojení JMS ve volném fondu. Pokud existuje, přesune se do aktivního fondu a bude předána aplikaci. Jinak se vytvoří nové připojení JMS , vloží se do aktivního fondu a vrátí se do aplikace. Maximální počet připojení, která 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 JMS a jejím zavření je připojení přesunuto z aktivního fondu do volného fondu, kde je k dispozici pro opětovné použití. Vlastnost fondu připojení **Unused timeout** definuje, jak dlouho může připojení JMS zůstat ve volném fondu, než bude odpojeno. Výchozí hodnota této vlastnosti je 1800 sekund (30 minut).

Při prvním vytvoření připojení JMS se spustí konverzace mezi WebSphere Application Server a IBM MQ . Konverzace zůstává aktivní, dokud není připojení uzavřeno, když je překročena hodnota vlastnosti **Unused timeout** pro volný fond.

Jedna konverzace pro každou relaci JMS , která byla vytvořena z připojení JMS

Každé připojení JMS vytvořené z továrny připojení poskytovatele systému zpráv IBM MQ má přidružený fond relací JMS . Fondy relací pracují stejným způsobem jako fondy připojení. Maximální počet JMS relací, které lze vytvořit z jednoho připojení produktu JMS , je určen vlastností fondu relací továrny připojení **Maximum connections**. Výchozí hodnota této vlastnosti je 10.

Konverzace se spustí při prvním vytvoření relace JMS . Konverzace zůstane aktivní, dokud se relace JMS nezavře, protože zůstala ve volném fondu déle, než je hodnota vlastnosti **Unused timeout** pro fond relací.

Výpočet hodnoty pro vlastnost SHARECNV

Maximální počet konverzací z jedné továrny připojení do produktu 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 pro umožnění tohoto počtu konverzací, lze vypracovat pomocí následujícího výpočtu:

```
Maximum number of channel instances =
    Maximum number of conversations / SHARECNV for the channel being used
```

Jakýkoli zbytek z tohoto výpočtu lze zaokrouhlit nahoru.

Pro jednoduchou továrnu připojení, která používá výchozí hodnotu pro vlastnosti fondu připojení **Maximum connections** a fondu relací **Maximum connections** , je maximální počet konverzací, které mohou existovat mezi WebSphere Application Server a IBM MQ pro tuto továrnu připojení:

```
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, který má vlastnost **SHARECNV** nastavenou 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 objektů typu message-driven bean, které jsou konfigurovány pro použití specifikace aktivace, mají aktivní konverzace pro specifikaci aktivace pro monitorování cíle JMS a pro každou relaci serveru používanou ke spuštění instance objektu typu message-driven bean pro zpracování zpráv.

Pro aplikace objektů typu message-driven bean, které jsou konfigurovány pro použití specifikace aktivace, jsou aktivní následující konverzace:

- Jedna konverzace pro specifikaci aktivace pro monitorování vhodných zpráv v cíli JMS . Tato konverzace se spustí, jakmile se spustí 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 pro zpracování zpráv.

Rozšířená vlastnost specifikace aktivace **Maximum server sessions** určuje maximální počet relací serveru, které mohou být současně aktivní pro danou specifikaci aktivace. Tato vlastnost má výchozí hodnotu 10. Relace serveru jsou vytvářeny podle potřeby a jsou uzavřeny, 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 spustí při vytvoření relace serveru a zastaví se buď při zastavení specifikace aktivace, nebo při vypršení časového limitu relace serveru.

To znamená, že maximální počet konverzací z jedné specifikace aktivace do produktu IBM MQ lze vypočítat pomocí následujícího vzorce:

```
Maximum number of conversations = Maximum server sessions + 1
```

Počet instancí kanálu, které byly vytvořeny pro umožnění tohoto počtu konverzací, lze nalézt pomocí následujícího výpočtu:

```
Maximum number of channel instances =
    Maximum number of conversations / SHARECNV for the channel being used
```

Jakýkoli zbytek z tohoto výpočtu lze zaokrouhlit nahoru.

Pro jednoduchou specifikaci aktivace, která používá výchozí hodnotu pro vlastnost **Maximum server sessions** , se maximální počet konverzací, které mohou existovat mezi WebSphere Application Server a IBM MQ pro tuto specifikaci aktivace, vypočítá takto:

```
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 vlastnost **SHARECNV** nastavenou na hodnotu 10, počet vytvořených instancí kanálu se vypočítá 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 používané aplikacemi objektů typu message-driven bean vytvářejí konverzace pro každou relaci serveru. Jeden monitoruje místo určení pro vhodné zprávy a druhý spouští instanci objektu typu message-driven bean pro zpracování zpráv. Počet konverzací pro každý port modulu listener lze vypočítat z maximálního počtu relací.

Standardně budou porty modulu listener spuštěny v režimu ASF jako součást specifikace 1.1 , která definuje mechanizmus, který by měly aplikační servery používat ke zjišťování zpráv a jejich doručování objektům typu message-driven bean ke zpracování. Aplikace objektů typu message-driven bean, které jsou nastaveny tak, aby používaly porty modulu listener v tomto výchozím režimu operace, vytvářejí konverzace:

Jedna konverzace pro port modulu listener pro monitorování místa určení pro vhodné zprávy
: Porty modulu listener jsou konfigurovány pro použití továrny připojení JMS . Při spuštění portu modulu listener je učiněn požadavek na připojení JMS z volného fondu továrny připojení. Po zastavení portu modulu listener je připojení vráceno do volného fondu. Další informace o způsobu použití fondu připojení a o tom, jak to ovlivňuje počet konverzací s produktem IBM MQ, naleznete v části “Továrny připojení JMS” na stránce 489.

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 pro daný port modulu listener současně aktivní. Tato vlastnost má výchozí hodnotu 10. Relace serveru jsou vytvářeny podle potřeby a využívají relace JMS převzaté z fondu relací přidruženého k připojení JMS , které port modulu listener používá.

Pokud byla relace serveru nečinná po dobu určenou přizpůsobenou vlastností **SERVER.SESSION.POOL.UNUSED.TIMEOUT**služby listener pro zprávy, relace se zavře a použitá relace JMS se vrátí do fondu volných relací. Relace JMS zůstane ve fondu volných relací, dokud nebude potřeba, nebo bude uzavřena, protože byla nečinná ve fondu volných relací déle, než je hodnota vlastnosti **Unused timeout** fondu relací.

Další informace o tom, jak se používá fond relací a jak se spravují konverzace mezi WebSphere Application Server a IBM MQ , viz “Továrny připojení JMS” na stránce 489.

Další informace o přizpůsobené vlastnosti služby listener pro zprávy **SERVER.SESSION.POOL.UNUSED.TIMEOUT**, viz téma Monitorování fondů relací serveru pro porty modulu listener v dokumentaci produktu WebSphere Application Server .

Výpočet maximálního počtu konverzací z jednoho portu modulu listener na IBM MQ

Maximální počet konverzací z jednoho portu modulu listener na IBM MQ můžete vypočítat pomocí následujícího vzorce:

```
Maximum number of conversations = Maximum sessions + 1
```

Počet instancí kanálu, které budou vytvořeny pro umožnění tohoto počtu konverzací, lze vypracovat pomocí následujícího výpočtu:

```
     Maximum number of channel instances =
        Maximum number of conversations / SHARECNV for the channel being used
```

Jakýkoli zbytek z tohoto výpočtu lze zaokrouhlit nahoru.

Pro jednoduchý port modulu listener, který používá výchozí hodnotu pro vlastnost **Maximum sessions** , se maximální počet konverzací, které mohou existovat mezi WebSphere Application Server a IBM MQ pro tento port modulu listener, vypočítá takto:

```
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 vlastnost **SHARECNV** nastavenou na hodnotu 10, bude počet instancí kanálu, které budou vytvořeny, vypočítán 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 non-ASF (Application Server Facilities)

Porty modulu listener spuštěné v jiném režimu než ASF lze konfigurovat tak, aby monitorovaly cíl fronty a cíl tématu pomocí relací serveru. Relace serveru mohou mít více konverzací, jejichž maximální počet lze v každém případě vypočítat.

Porty modulu listener lze konfigurovat tak, aby byly spouštěny v jiném režimu než ASF, což mění způsob, jakým porty modulu listener monitorují cíle JMS . Aplikace objektů typu message-driven bean s použitím portů modulu listener v provozním režimu, který není ASF, vytvoří konverzaci pro každou relaci serveru používanou ke spuštění instance objektu typu message-driven bean pro zpracování zpráv. Vlastnost portu modulu listener **maximum relací** určuje maximální počet relací serveru, které mohou být pro daný port modulu listener současně aktivní. Výchozí hodnota této vlastnosti je 10.

Při spuštění v jiném režimu než ASF port modulu listener, který monitoruje cíl fronty, automaticky vytvoří počet relací serveru určený vlastností portu modulu listener **Maximální počet relací**. Všechny tyto relace serveru využívají relace JMS převzaté z fondu relací přidruženého k připojení JMS , které port modulu listener používá, a průběžně monitorují cíl JMS pro vhodné zprávy.

Pokud je port modulu listener konfigurován pro monitorování místa určení tématu, hodnota **Maximální počet relací** se ignoruje a použije se jedna relace serveru.

Relace serveru používané portem modulu listener spuštěným v jiném režimu než ASF zůstávají aktivní, dokud není port modulu listener zastaven. V tomto bodě jsou relace JMS , které byly použity, vráceny do volného fondu fondu relací pro připojení produktu JMS , které port modulu listener používal.

Další informace o tom, jak se používá fond relací a jak se spravují konverzace mezi WebSphere Application Server a IBM MQ , viz “Továrny připojení JMS” na stránce 489.

Další informace o režimu ASF a jiném režimu ASF s produktem WebSphere Application Servera o tom, jak nakonfigurovat porty modulu listener tak, aby používaly jiný režim než ASF, naleznete v tématu Zpracování zpráv v režimu ASF a jiném režimu ASF.

Výpočet maximálního počtu konverzací při monitorování cíle fronty

Maximální počet konverzací z jednoho portu modulu listener spuštěných v jiném režimu než ASF a monitorujících cíl 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, lze nalézt pomocí následujícího výpočtu:

```
Maximum number of channel instances =
   Maximum number of conversations / SHARECNV for the channel being used
```

Jakýkoli zbytek 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 pomocí kanálu, který má vlastnost **SHARECNV** nastavenou na hodnotu 10, počet vytvořených instancí kanálu se vypočítá 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 jiném režimu než ASF a konfigurovaný pro monitorování místa určení tématu je počet konverzací z portu modulu listener do IBM MQ následující:

```
Maximum number of conversations = 1
```

Počet instancí kanálu, které budou vytvořeny pro umožnění tohoto počtu konverzací, lze nalézt pomocí následujícího výpočtu:

```
Maximum number of channel instances =
   Maximum number of conversations / SHARECNV for the channel being used
```

Jakýkoli zbytek 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 místa určení tématu, je maximální počet konverzací, které mohou existovat mezi WebSphere Application Server a IBM MQ pro tento port modulu listener, následující:

```
Maximum number of conversations = Maximum sessions
```

Příklad:

```
= 10
```

Pokud se tento port modulu listener připojuje k produktu IBM MQ pomocí kanálu, který má vlastnost **SHARECNV** nastavenou na hodnotu 10, počet vytvořených instancí kanálu se vypočítá takto:

```
Maximum number of channel instances =
   Maximum number of conversations / SHARECNV for the channel being used
```

Příklad:

```
= 10 / 10
= 1
```

Konfigurace aliasů ověřování pro zabezpečení WebSphere Application Server připojení k IBM MQ

Aliasy ověřování se mapují na kombinaci jména uživatele a hesla, kterou lze použít k zabezpečení WebSphere Application Server připojení k IBM MQ. Můžete konfigurovat továrnu připojení s aliasem ověřování.

Použití aliasů ověřování s podnikovými aplikacemi

Když se podniková aplikace spuštěná uvnitř produktu WebSphere Application Server pokusí vytvořit JMS připojení k IBM MQ, aplikace vyhledá definici továrny připojení poskytovatele systému zpráv IBM MQ z úložiště Java Naming Directory Interface (JNDI) aplikačního serveru.

Je-li definice továrny připojení poskytovatele systému zpráv IBM MQ umístěna v úložišti JNDI aplikačního serveru, je volána jedna z následujících metod:

- `ConnectionFactory.createConnection()`
- `ConnectionFactory.createConnection(String username, String password)`

Pokud byla továrna připojení konfigurována s definovaným aliasem ověřování J2C , lze jméno uživatele a heslo v aliasu ověřování použít při použití továrny připojení k vytvoření připojení k produktu IBM MQ .

Továrny připojení a aliasy ověřování

Továrny připojení poskytovatele systému zpráv IBM MQ obsahují informace o způsobu připojení ke správcům front produktu IBM MQ . Podnikové aplikace spuštěné uvnitř produktu WebSphere Application Server mohou pomocí továren připojení vytvářet JMS připojení k produktu IBM MQ.

Produkt WebSphere Application Server ukládá definice továren připojení do úložiště, ke kterému lze přistupovat pomocí konzoly JNDI. Při vytvoření továrny připojení je továrně připojení poskytnut název JNDI , který ji jedinečně identifikuje v oboru aplikačního serveru (buňka, uzel nebo obor serveru), v němž byla definována.

Například továrna připojení poskytovatele systému zpráv IBM MQ definovaná v oboru buňky WebSphere Application Server obsahuje informace o tom, jak se připojit ke správci front (`myQM`) pomocí přenosu BINDINGS. Této továrně připojení je udělen JNDI název `jms/myCF` , který ji jedinečně identifikuje.

Továrny připojení lze také nakonfigurovat tak, aby používaly alias ověřování. Aliasy ověřování se mapují na kombinaci jména uživatele a hesla. V závislosti na tom, jak se používá továrna připojení, může nebo nemusí být jméno uživatele a heslo v aliasu ověřování při vytvoření připojení produktu JMS zadáno do adresáře IBM MQ .

Důležité: Prior to IBM MQ 8.0, the default IBM MQ Object Authority Manager (OAM) performed an authorization check, only to ensure that the user name passed down to IBM MQ, when a connection is made, had the authority to access the queue manager.

Nebyly provedeny žádné kontroly pro ověření zadaného hesla. Chcete-li provést kontrolu ověření a ověřit, zda se identifikátor uživatele a heslo shodují, musíte napsat uživatelskou proceduru zabezpečení kanálu IBM MQ . Podrobnosti o tom, jak to provést, naleznete v části “Uživatelské programy zabezpečení kanálu” na stránce 936.

V produktu IBM MQ 8.0kontroluje správce front kromě jména uživatele také heslo.

Použití továrny připojení

Následující témata obsahují informace o použití továrny připojení pomocí přímého a nepřímého vyhledání:

- “Použití továrny připojení prostřednictvím přímého vyhledávání” na stránce 498
- “Použití továrny připojení prostřednictvím nepřímého vyhledávání” na stránce 499

Použití přenosu CLIENT

Továrny připojení, které jsou konfigurovány pro použití transportu CLIENT, musí určovat, který kanál připojení serveru IBM MQ (SVRCONN) budou používat pro připojení ke správci front.

Pokud vlastnost MCAUSER (identifikátor uživatele agenta kanálu IBM MQ) zůstává pro kanál, pro jehož použití byla továrna připojení konfigurována, prázdná, lze továrnu připojení použít buď s přímým vyhledáním, nebo s nepřímým vyhledáním.

Je-li vlastnost MCAUSER nastavena na identifikátor uživatele, předá se tento identifikátor uživatele produktu IBM MQ , když se továrna připojení používá k vytvoření připojení k produktu IBM MQ, bez ohledu na to, zda podniková aplikace používá přímé nebo nepřímé vyhledání.

souhrnné tabulky

Následující tabulky shrnují, které identifikátory uživatelů jsou při použití transportu BINDINGS a při použití transportu CLIENT sledovány směrem dolů do adresáře IBM MQ :

Tabulka 71. 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 je v toku dolů do adresáře IBM MQ.	Identifikátor uživatele a heslo, které byly předány do metody `ConnectionFactory.createConnection(String username, String password)` , jsou protečeny do adresáře IBM MQ.

Tabulka 71. Režim BINDINGS (pokračování)

Konfigurace	**Volání aplikace `ConnectionFactory.createConnection()`**	**Volání aplikace `ConnectionFactory.createConnection(String username, String password)`**
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 je v toku dolů do adresáře IBM MQ.	Identifikátor uživatele a heslo, které byly předány do metody `ConnectionFactory.createConnection(String username, String password)` , jsou protečeny do adresáře 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 továrnu připojení jsou protékány do adresáře IBM MQ.	Identifikátor uživatele a heslo zadané v aliasu ověřování pro továrnu připojení jsou protékány do adresáře 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 ověřování, pro jehož použití byla aplikace konfigurována, jsou uvedeny v toku do adresáře IBM MQ.	Identifikátor uživatele a heslo zadané v aliasu ověřování, pro jehož použití byla aplikace konfigurována, jsou uvedeny v toku do adresáře IBM MQ.

Tabulka 72. Režim KLIENT

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 tak, aby používala kanál IBM MQ , jehož vlastnost MCAUSER není nastavena.	Identifikátor uživatele pro proces aplikačního serveru je v toku dolů do adresáře IBM MQ.	Identifikátor uživatele a heslo, které byly předány do metody `ConnectionFactory.createConnection(String username, String password)` , jsou protečeny do adresáře IBM MQ.
Deskriptor implementace aplikace neobsahuje odkaz na prostředek pro továrnu připojení a továrna připojení je konfigurována pro použití kanálu IBM MQ , který má vlastnost MCAUSER nastavenou na identifikátor uživatele.	Identifikátor uživatele určený vlastností MCAUSER v kanálu IBM MQ , pro který je továrna připojení konfigurována, je toku dolů do adresáře IBM MQ.	Identifikátor uživatele určený vlastností MCAUSER v kanálu IBM MQ , pro který je továrna připojení konfigurována, je toku dolů do adresáře IBM MQ.

Tabulka 72. Režim KLIENT (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** nastavenou na hodnotu *Aplikace* , a továrna připojení je nakonfigurována tak, aby používala kanál IBM MQ , který má nenastavenou vlastnost MCAUSER.	Identifikátor uživatele pro proces aplikačního serveru je v toku dolů do adresáře IBM MQ.	Identifikátor uživatele a heslo, které byly předány do metody `ConnectionFactory.createConnection(String username, String password)` , jsou protečeny do adresáře IBM MQ.
Deskriptor implementace aplikace obsahuje odkaz na prostředek pro továrnu připojení, která má vlastnost **res-auth** nastavenou na hodnotu *Aplikace* , a továrna připojení je konfigurována pro použití kanálu IBM MQ , který má vlastnost MCAUSER nastavenou na identifikátor uživatele.	Identifikátor uživatele určený vlastností MCAUSER v kanálu IBM MQ , pro jehož použití je továrna připojení konfigurována, je protékán směrem dolů do adresáře IBM MQ.	Identifikátor uživatele určený vlastností MCAUSER v kanálu IBM MQ , pro jehož použití je továrna připojení konfigurována, je protékán směrem dolů do adresáře 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 nakonfigurována tak, aby používala kanál IBM MQ , který má nenastavenou vlastnost MCAUSER.	Identifikátor uživatele a heslo zadané v aliasu ověřování pro továrnu připojení jsou protékány do adresáře IBM MQ.	Identifikátor uživatele a heslo zadané v aliasu ověřování pro továrnu připojení jsou protékány do adresáře 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 nakonfigurována tak, aby používala kanál IBM MQ , který má vlastnost MCAUSER nastavenou na identifikátor uživatele.	Identifikátor uživatele určený vlastností MCAUSER v kanálu IBM MQ , pro jehož použití je továrna připojení konfigurována, je protékán směrem dolů do adresáře IBM MQ.	Identifikátor uživatele určený vlastností MCAUSER v kanálu IBM MQ , pro jehož použití je továrna připojení konfigurována, je protékán směrem dolů do adresáře IBM MQ.

Tabulka 72. Režim KLIENT (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`** nastavenou na hodnotu "*Kontejner* , a aplikace byla konfigurována s aliasem ověřování a továrna připojení je konfigurována tak, aby používala kanál IBM MQ s nenastavenou vlastností MCAUSER.	Identifikátor uživatele a heslo zadané v aliasu ověřování, pro jehož použití byla aplikace konfigurována, jsou uvedeny v toku do adresáře IBM MQ.	Identifikátor uživatele a heslo zadané v aliasu ověřování, pro jehož použití byla aplikace konfigurována, jsou uvedeny v toku do adresáře IBM MQ.
Deskriptor implementace aplikace obsahuje odkaz na prostředek pro továrnu připojení, která má vlastnost **`res-auth`** nastavenu na hodnotu *Kontejner* , a aplikace byla konfigurována s aliasem ověřování a továrna připojení je konfigurována tak, aby používala kanál IBM MQ , který má vlastnost MCAUSER nastavenou na identifikátor uživatele.	Identifikátor uživatele určený vlastností MCAUSER v kanálu IBM MQ , pro jehož použití je továrna připojení konfigurována, je protékán směrem dolů do adresáře IBM MQ.	Identifikátor uživatele určený vlastností MCAUSER v kanálu IBM MQ , pro jehož použití je továrna připojení konfigurována, je protékán směrem dolů do adresáře IBM MQ.

Použití továrny připojení prostřednictvím přímého vyhledávání

Po definování továrny připojení poskytovatele systému zpráv IBM MQ může podniková aplikace vyhledat definici továrny připojení a použít ji k vytvoření připojení JMS ke správci front IBM MQ . To lze provést prostřednictvím přímého vyhledání.

Chcete-li použít přímé vyhledávání, podniková aplikace se připojí k úložišti JNDI aplikačního serveru provedením následujícího volání metody:

```
InitialContext ctx = new InitialContext();
```

Po připojení k úložišti JNDI identifikuje podniková aplikace definici továrny připojení pomocí názvu JNDI továrny připojení, a to následujícím způsobem:

```
ConnectionFactory cf = (ConnectionFactory) ctx.lookup("jms/myCF");
```

Notes:

- Vývojář aplikací musí při vývoji podnikové aplikace znát název JNDI požadované továrny připojení. Vzhledem k tomu, že název JNDI je v aplikaci pevně zakódován, musíte v případě, že se název JNDI změní, přepsat a znovu implementovat aplikaci.
- Je-li použita definice továrny připojení tímto způsobem, jméno uživatele a heslo zadané v aliasu ověřování (pro jehož použití byla továrna připojení konfigurována) se netečou do adresáře IBM MQ. To má zabránit tomu, aby neautorizované aplikace identifikovaly továrnu připojení a mohly ji používat pro připojení k zabezpečeným systémům IBM MQ .

Jméno uživatele a heslo, které jsou v toku dolů do adresáře IBM MQ , závisí na metodě použité k vytvoření připojení JMS z továrny připojení.

Pokud aplikace vytvoří připojení JMS pomocí 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.

Alternativně může aplikace vytvořit připojení JMS voláním metody:

```
ConnectionFactory.createConnection(String username, String password)
```

Pokud aplikace provedla přímé vyhledání továrny připojení a poté tuto metodu zavolala, jméno uživatele a heslo, které byly předány do metody `createConnection()` , se přenesou do adresáře IBM MQ.

Důležité: Před IBM MQ 8.0, IBM MQ zpracoval kontrolu autorizace, pouze aby se ujistil, že jméno uživatele, které bylo v toku dolů, mělo oprávnění pro přístup ke správci front.

Nebyly provedeny žádné kontroly hesla. Chcete-li provést kontrolu ověření a ověřit platnost jména uživatele a hesla, musí být zapsána uživatelská procedura zabezpečení kanálu IBM MQ . Podrobnosti o tom, jak to provést, naleznete v části “Uživatelské programy zabezpečení kanálu” na stránce 936.

V produktu IBM MQ 8.0kontroluje správce front kromě jména uživatele také heslo.

Použití továrny připojení prostřednictvím nepřímého vyhledávání

Pokud při psaní podnikové aplikace není znám název JNDI továrny připojení nebo pokud má být aplikace nainstalována na různé aplikační servery s použitím jiné továrny připojení s jiným názvem produktu JNDI (v závislosti na tom, na který aplikační server je nainstalována), lze továrnu připojení vyhledat pomocí odkazu na prostředek. To lze provést prostřednictvím nepřímého vyhledávání.

Příklad

Namísto přímého vyhledávání továrny připojení pomocí produktu `jms/myCF`obsahuje podniková aplikace odkaz na prostředek lokální JNDI název: `jms/myResourceReferenceCF`.

Chcete-li použít tento název JNDI , připojí se aplikace k úložišti JNDI aplikačního serveru stejným způsobem, jako kdyby aplikace prováděla přímé vyhledání:

```
InitialContext ctx = new InitialContext();
```

Namísto přímé identifikace `jms/myCF` nyní aplikace identifikuje název JNDI odkazu na prostředek:

```
ConnectionFactory cf = (ConnectionFactory) ctx.lookup("java:comp/env/jms/
myResourceReferenceCF");
```

Potřebujete předponu `java:comp/env` pro lokální název JNDI , abyste informovali aplikační server, že podniková aplikace provádí nepřímé vyhledání.

Když je aplikace implementována, uživatel mapuje JNDI název odkazu na prostředek `jms/myResourceReferenceCF` na název rozhraní JNDI továrny připojení, kterou již aplikace vytvořila: `jms/myCF`.

Když je aplikace spuštěna, vyhledá továrnu připojení JMS pomocí lokálního názvu JNDI , na který je aplikační server mapován: `jms/myCF`. Tuto továrnu připojení pak aplikace používá k vytvoření připojení k produktu IBM MQ.

Aliasy ověřování a nepřímá vyhledávání

Odkaz na prostředek také umožňuje definovat další vlastnosti, které mění chování poskytované továrny připojení. Jednou z vlastností odkazu na prostředek je **res-auth**. Hodnota této vlastnosti určuje, zda má podniková aplikace při vytváření připojení k produktu IBM MQ (pokud byl definován alias ověřování) používat alias ověřování továrny připojení, na kterou je mapován odkaz na prostředek, nebo zda aplikace zadává své vlastní jméno uživatele a heslo.

Výchozí hodnota této vlastnosti je *Aplikace*. To znamená, že jméno uživatele a heslo, které jsou při vytvoření připojení JMS do správce front tečeny, je určeno samotnou aplikací. Alias ověřování továrny připojení, na kterou je odkaz na prostředek mapován, není použit.

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 parametr **res-auth** je nastaven na hodnotu *Aplikace*, výchozí identita uživatele bude tečena do adresáře 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 parametr **res-auth** je nastaven na hodnotu *Aplikace*, jméno uživatele a heslo předané metodě se odešlou na adresu IBM MQ.

Chcete-li použít alias ověřování definovaný v továrně připojení, na kterou je mapován odkaz na prostředek při vytváření připojení, musíte nastavit vlastnost **res-auth** na hodnotu *Kontejner*. Když aplikace vytvoří připojení JMS , použijí se podrobnosti o aliasu ověřování, i když volání `createConnection` uvádí jméno uživatele a heslo.

Potlačení aliasu ověřování při použití nepřímého vyhledávání

Pokud aplikace používá odkaz na prostředek, který má vlastnost **res-auth** nastavenou na *Kontejner*, můžete přepsat alias ověřování, který se používá při vytváření připojení produktu JMS .

Chcete-li potlačit alias ověřování, musí odkaz na prostředek obsahovat další vlastnost s názvem **authDataAlias**, která se mapuje na existující alias ověřování, který již byl vytvořen v prostředí aplikačního serveru, do kterého bude aplikace implementována. Tuto vlastnost můžete zadat pro všechny odkazy na prostředky, které jsou vytvořeny pomocí nástrojů Rational poskytovaných produktem IBM.

Pomocí této metody můžete při použití továrny připojení JMS , která byla vyhledána nepřímo, použít jiný alias ověřování. Pokud zadaný alias ověřování neexistuje, lze po instalaci podnikové aplikace zadat nový. Další informace viz *Odkazy na prostředky* v dokumentaci k 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 typu message-driven bean při použití klastrů WebSphere Application Server

Při použití aplikací objektů typu message-driven bean implementovaných v klastru WebSphere Application Server 7.0 a WebSphere Application Server 8.0 a nakonfigurovaných ke spuštění v normálním režimu poskytovatele systému zpráv IBM MQ zpracovává jeden z členů klastru většinu zpráv. Můžete vyvážit pracovní zátěž členů klastru, abyste rozdělili zpracování zpráv na více než jednoho člena klastru.

Produkt IBM MQ obsahuje funkci s názvem **Asynchronous consume**, která umožňuje aplikacím asynchronně přijímat zprávy z fronty pomocí rozhraní API s názvem **MQCB** a **MQCTL**.

Aplikace objektů typu message-driven bean spuštěné v systémech WebSphere Application Server 7.0 a WebSphere Application Server 8.0, které používají normální režim poskytovatele systému zpráv IBM MQ , budou tuto funkci automaticky využívat. Když se aplikace spustí, nastaví asynchronního spotřebitele v cíli JMS , který byly nakonfigurovány k monitorování voláním funkce **MQCB**. Poté je voláno rozhraní API **MQCTL** , které označuje, že aplikace je připravena přijímat zprávy z místa určení JMS .

Když byly aplikace objektů typu message-driven bean implementovány do klastru WebSphere Application Server , každý člen klastru nastaví asynchronního spotřebitele pro místo určení JMS , které objekt typu message-driven bean monitoruje pro zprávy. Správce front IBM MQ , který je hostitelem místa určení JMS , je poté odpovědný za oznámení členovi klastru v případě, že v místě určení JMS existuje vhodná zpráva, kterou má zpracovat.

Když se produkt WebSphere Application Server připojuje ke správci front IBM MQ , zprávy, které dorazí do cíle systému JMS , budou distribuovány rovnoměrněji všem asynchronním spotřebitelům, kteří byli registrováni v tomto cíli systému JMS . Pro aplikace objektů typu message-driven bean implementované uvnitř klastru WebSphere Application Server 7.0 a WebSphere Application Server 8.0 to znamená, že zprávy budou distribuovány rovnoměrněji mezi členy klastru.

Související úlohy

Konfigurace vlastnosti JMS **PROVIDERVERSION**

Použití balíku záhlaví IBM MQ

Balík IBM MQ Headers poskytuje sadu pomocných rozhraní a tříd, které můžete použít k manipulaci se záhlavími IBM MQ zprávy. Obvykle používáte balík záhlaví IBM MQ , protože chcete provádět administrativní služby pomocí příkazového serveru (pomocí zpráv PCF (Programmable Command Format)).

Informace o této úloze

Balík záhlaví IBM MQ je umístěn v balících `com.ibm.mq.headers` a `com.ibm.mq.headers.pcf` . Tento prostředek můžete použít pro obě dvě alternativní rozhraní API, která produkt IBM MQ poskytuje pro použití v aplikacích Java :

- IBM MQ classes for Java (také označováno jako IBM MQ Základní Java).
- IBM MQ classes for Java Message Service (IBM MQ classes for JMS, také nazýváno IBM MQ JMS).

IBM MQ Základní Java aplikace obvykle manipulují s objekty MQMessage a třídy podpory Headers mohou s těmito objekty přímo interagovat, protože přirozeně rozumí rozhraním IBM MQ Základní Java .

V produktu IBM MQ JMSje informačním obsahem zprávy obvykle řetězec nebo objekt bajtového pole, s nímž lze manipulovat pomocí proudů DataInput a DataOutput . Balík IBM MQ Headers lze použít k interakci s těmito datovými proudy a je vhodný pro manipulaci se zprávami produktu MQ , které jsou odesílány a přijímány aplikacemi produktu IBM MQ JMS .

Ačkoli tedy balík IBM MQ Headers obsahuje odkazy na balík IBM MQ Základní Java , je také určen pro použití v aplikacích IBM MQ JMS a je vhodný pro použití v prostředích Java Platform, Enterprise Edition (Java EE).

Typickým způsobem, jak můžete použít balík záhlaví IBM MQ , je manipulovat se zprávami administrace v programovatelném formátu příkazů (PCF), například z následujících důvodů:

- Přístup k podrobnostem o prostředku IBM MQ .
- Monitorování hloubky fronty.
- Blokování přístupu k frontě.

Pomocí zpráv PCF s rozhraním API produktu IBM MQ JMS lze tento druh administrace prostředků zaměřených na aplikace provádět z aplikací Java EE , aniž by bylo nutné používat rozhraní API IBM MQ Base Java .

Procedura

- Chcete-li použít balík IBM MQ Záhlaví pro manipulaci se záhlavími zpráv pro IBM MQ classes for Java, viz “Použití s IBM MQ classes for Java” na stránce 502.
- Chcete-li použít balík IBM MQ Záhlaví pro manipulaci se záhlavími zpráv pro IBM MQ classes for JMS, viz “Použití s IBM MQ classes for JMS” na stránce 502.

Použití s IBM MQ classes for Java

Aplikace IBM MQ classes for Java obvykle manipulují s objekty MQMessage a třídy podpory Headers mohou přímo interaktivně spolupracovat s těmito objekty, protože nativně rozumí rozhraním IBM MQ classes for Java .

Informace o této úloze

Produkt IBM MQ poskytuje některé ukázkové aplikace, které demonstrují, jak používat balík IBM MQ Headers s rozhraním IBM MQ Base Java API (IBM MQ classes for Java).

Ukázky ukazují dvě věci:

- Jak vytvořit zprávu PCF pro provedení administrativní akce a analýzu zprávy odpovědi.
- Jak odeslat tuto zprávu PCF pomocí IBM MQ classes for Java.

V závislosti na používané platformě jsou tyto ukázky nainstalovány v adresáři `pcf` v adresáři `samples` nebo `tools` vaší instalace produktu IBM MQ (viz “Instalační adresáře pro IBM MQ classes for Java” na stránce 343).

Postup

1. Vytvořte zprávu PCF k provedení administrativní akce a analyzujte zprávu odpovědi.
2. Odešlete tuto zprávu PCF pomocí konzoly IBM MQ classes for Java.

Související pojmy

“Obsluha záhlaví zpráv IBM MQ pomocí IBM MQ classes for Java” na stránce 369
Jsou poskytnuty třídy Java představující různé typy záhlaví zprávy. K dispozici jsou také dvě pomocné třídy.

“Zpracování zpráv PCF pomocí IBM MQ classes for Java” na stránce 374
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 s IBM MQ classes for JMS, proveďte stejné nezbytné kroky jako pro IBM MQ classes for Java. Zprávu PCF lze vytvořit a odpověď analyzovat přesně stejným způsobem pomocí balíku záhlaví IBM MQ a stejného ukázkového kódu jako pro IBM MQ classes for Java.

Informace o této úloze

Chcete-li odeslat zprávu PCF pomocí rozhraní API IBM MQ , informační obsah zprávy musí být zapsán do JMS Bajtových zpráv a odeslán pomocí standardních rozhraní API JMS . Jediným aspektem je, že zpráva nesmí obsahovat JMS RFH2 ani žádná jiná záhlaví se specifickými hodnotami v deskriptoru MQMD.

Chcete-li odeslat zprávu PCF, postupujte takto. Způsob, jakým je zpráva PCF vytvořena, a informace jsou extrahovány ze zprávy odpovědi, je stejný jako pro IBM MQ classes for Java (viz “Použití s IBM MQ classes for Java” na stránce 502).

Postup

1. Vytvořte cíl fronty JMS , který představuje SYSTEM.ADMIN.COMMAND.QUEUE.

Aplikace IBM MQ JMS odesílají zprávy PCF do SYSTEM.ADMIN.COMMAND.QUEUEa potřebujete přístup k objektu cíle JMS, který představuje tuto frontu. Cíl musí mít nastaveny následující vlastnosti:

```
WMQ_MQMD_WRITE_ENABLED = YES
WMQ_MESSAGE_BODY = MQ
```

Používáte-li produkt WebSphere Application Server, musíte tyto vlastnosti definovat jako přizpůsobené vlastnosti v místě určení.

Chcete-li vytvořit cíl programově z 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 JMS Bajtovou zprávu obsahující správné hodnoty MQMD.

   JMS Bajtová zpráva musí být vytvořena a do ní musí být zapsána zpráva PCF. Je třeba vytvořit frontu odpovědí, ale tato fronta nemusí mít žádná specifická nastavení.

   Následující ukázkový úsek kódu ukazuje, jak vytvořit JMS Bajtovou zprávu a zapsat do ní objekt com.ibm.mq.headers,pcf.PCFMessage . Objekt PCFMessage (pcfCmd) byl již 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 JMS .
4. Převeďte zprávu odpovědi na zprávu PCF pro 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 139

Zprávy JMS se skládají ze záhlaví, vlastností a těla. JMS definuje pět typů těla zprávy.

IBM i Nastavení IBM MQ na systému IBM i pomocí Java a JMS

Tato kolekce témat poskytuje přehled o tom, jak nastavujete a testujete IBM MQ pomocí Java a JMS na systému IBM i pomocí příkazů CL nebo prostředí qshell.

Poznámka:

- Z IBM MQ 8.0, `ldap.jar`, `jndi.jar` a `jta.jar` jsou součástí sady JDK.
- V 9.3.0 JM 3.0 V 9.3.0 Od IBM MQ 9.3.0, Jakarta Messaging 3.0 je podporován pro vývoj nových aplikací. Produkt IBM MQ 9.3.0 nadále podporuje produkt JMS 2.0 pro existující aplikace. Použití rozhraní API Jakarta Messaging 3.0 a rozhraní API JMS 2.0 ve stejné aplikaci není podporováno. Další informace naleznete v tématu Použití tříd IBM MQ pro systém zpráv JMS/Jakarta.

Použití CL příkazů

Parametr CLASSPATH, který jste nastavili, je určen pro testování s produktem MQ base Java, JMS s rozhraním JNDI a JMS bez rozhraní JNDI.

Pokud nepoužijete soubor `.profile` ve svém adresáři `/home/Userprofile` , budete muset níže nastavit proměnné prostředí na úrovni systému. Můžete zkontrolovat, zda jsou nastaveny pomocí příkazu **WRKENVVAR** .

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 parametr CLASSPATH nastaven, zadejte následující příkaz:

   JM 3.0

   ```
   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.mq.jakarta.client.jar
          :/QIBM/ProdData/mqm/java/lib/jms.jar
          :/QIBM/ProdData/mqm/java/lib/providerutil.jar
          :/QIBM/ProdData/mqm/java/lib/fscontext.jar:') LEVEL(*SYS)
   ```

   JMS 2.0

   ```
   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.mq.allclient.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. Pokud QIBM_MULTI_THREAD_NENÍ nastaveno, zadejte následující příkaz:

   ```
   ADDENVVAR ENVVAR(QIBM_MULTI_THREADED) VALUE('Y') LEVEL(*SYS)
   ```

5. Není-li parametr QIBM_USE_DESCRIPTOR_STDIO nastaven, zadejte následující příkaz:

   ```
   ADDENVVAR ENVVAR(QIBM_USE_DESCRIPTOR_STDIO) VALUE('I') LEVEL(*SYS)
   ```

6. Není-li parametr QSH_REDIRECTION_TEXTDATA nastaven, zadejte následující příkaz:

```
ADDENVVAR ENVVAR(QSH_REDIRECTION_TEXTDATA) VALUE('Y') LEVEL(*SYS)
```

Použití prostředí qshell

Pokud používáte prostředí QSHELL, můžete nastavit `.profile` ve svém adresáři `/home/Userprofile`. Další informace viz dokumentace Qshell Interpreter (qsh).

V souboru `.profile`uveďte následující. Všimněte si, že příkaz CLASSPATH musí být na jednom řádku nebo musí být oddělen na různých řádcích pomocí znaku \ , jak je zobrazeno.

JM 3.0

```
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.mq.jakarta.client.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
```

JMS 2.0

```
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.mq.allclient.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
```

Zadáním příkazu **DSPLIBL**se ujistěte, že knihovna QMQMJAVA je v seznamu knihoven.

Pokud knihovna QMQMJAVA není v seznamu, přidejte ji pomocí následujícího příkazu: **ADDLIBLE LIB (QMQMJAVA)**

IBM i Testování IBM MQ na systému IBM i pomocí Java

Jak testujete produkt IBM MQ s produktem Java pomocí ukázkového programu MQIVP.

Testování IBM MQ základního Java

Proveďte následující postup:

1. Zadáním následujícího příkazu ověřte, zda je správce front spuštěn a zda je stav správce front AKTIVNÍ:

   ```
   WRKMQM MQMNAME(QMGRNAME)
   ```

2. Ověřte, že JAVA.CHANNEL byl vytvořen zadáním následujícího příkazu:

   ```
   WRKMQMCHL CHLNAME(JAVA.CHANNEL) CHLTYPE(*SVRCN) MQMNAME(QMGRNAME)
   ```

   a. Pokud je JAVA.CHANNEL neexistuje, zadejte následující příkaz:

```
CRTMQMCHL CHLNAME(JAVA.CHANNEL) CHLTYPE(*SVRCN) MQMNAME(QMGRNAME)
```

3. Zadáním příkazu **WRKMQMLSR** ověřte, zda je modul listener správce front spuštěn pro port 1414 nebo jakýkoli 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 cesta CLASSPATH, a poté zadejte příkaz **cd** následujícím způsobem:

```
cd /qibm/proddata/mqm/java/samples/wmqjava/samples
```

3. Spusťte program **java** zadáním následujícího příkazu:

```
java MQIVP
```

Můžete stisknout klávesu ENTER, když jste vyzváni k zadání:

- Typ připojení
- Adresa IP
- Název správce front

chcete-li použít výchozí hodnoty. Tím se ověří vazby produktu, které lze nalézt v knihovně QMQMJAVA.

Obdržíte výstup podobný následujícímu 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, 2024. 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

Obdržíte výstup podobný následujícímu 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, 2024. 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 systému IBM i pomocí JMS

Jak testujete produkt IBM MQ pomocí produktu JMS s rozhraním JNDI i bez něj

Testování produktu 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 AKTIVNÍ:

   ```
   WRKMQM MQMNAME(QMGRNAME)
   ```

2. Spusťte qshell z příkazového řádku zadáním příkazu **STRQSH** .
3. Pomocí příkazu **cd** změňte adresář takto:

   ```
   cd /qibm/proddata/mqm/java/bin
   ```

4. Spusťte skriptový soubor:

   ```
   IVTRun -nojndi [-m qmgrname]
   ```

Obdržíte výstup podobný následujícímu příkladu. Všimněte 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, 2024.
All Rights Reserved.
WebSphere MQ classes for Java 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 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 AKTIVNÍ:

   ```
   WRKMQM MQMNAME(QMGRNAME)
   ```

2. Zadáním následujícího příkazu ověřte, že je vytvořen kanál připojení serveru:

   ```
   WRKMQMCHL CHLNAME( SYSTEM.DEF.SVRCONN ) CHLTYPE(*SVRCN)
   MQMNAME(QMGRNAME)
   ```

3. Zadáním příkazu **WRKMQMLSR** ověřte, že je modul listener spuštěn pro správný port.
4. Spusťte qshell z příkazového řádku zadáním příkazu **STRQSH** .
5. Zadáním příkazu **export** ověřte, že je hodnota CLASSPATH správná.
6. Pomocí příkazu **cd** změňte adresář takto:

   ```
   cd /qibm/proddata/mqm/java/bin
   ```

7. Spusťte skriptový soubor:

   ```
   IVTRun -nojndi -client -m QMgrName -host hostname [-port port] [-channel channel]
   ```

Obdržíte výstup podobný následujícímu 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, 2024.
All Rights Reserved.
WebSphere MQ classes for Java 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 produktu IBM MQ JMS s rozhraním JNDI

Zadáním následujícího příkazu ověřte, zda je správce front spuštěn a zda je stav správce front AKTIVNÍ:

```
WRKMQM MQMNAME(QMGRNAME)
```

Použití ukázkového testovacího skriptu IVTRun

Proveďte následující postup:

1. Proveďte odpovídající změny v souboru JMSAdmin.config . Chcete-li upravit tento soubor, použijte příkaz **EDTF** (Upravit soubor) z příkazového řádku systému IBM i .

   ```
   EDTF '/qibm/proddata/mqm/java/bin/JMSAdmin.config'
   ```

   .

   a. Chcete-li použít LDAP pro Weblogic, odeberte komentář z:

   ```
   INITIAL_CONTEXT_FACTORY=com.sun.jndi.ldap.LdapCtxFactory
   ```

   b. Chcete-li použít LDAP pro WebSphere Application Server, odeberte komentář z:

   ```
   INITIAL_CONTEXT_FACTORY=com.ibm.ejs.ns.jndi.CNInitialContextFactory
   ```

   c. Chcete-li otestovat systém souborů, odeberte komentář z:

```
INITIAL_CONTEXT_FACTORY=com.sun.jndi.fscontext.RefFSContextFactory
```

d. Ujistěte se, že jste vybrali správnou hodnotu PROVIDER_URL, odebráním komentáře z příslušného řádku.

e. Označte všechny ostatní řádky jako komentář pomocí symbolu *#* .

f. Po dokončení všech změn stiskněte klávesy **F2=Save** a **F3=Exit**.

2. Spusťte qshell z příkazového řádku zadáním příkazu **STRQSH** .
3. Zadáním příkazu **export** ověřte, že je hodnota CLASSPATH správná.
4. Pomocí příkazu **cd** změňte adresář takto:

```
cd /qibm/proddata/mqm/java/bin
```

5. Spuštěním skriptu **IVTSetup** vytvořte spravované objekty (*MQQueueConnectionFactory* a *MQQueue*) zadáním příkazu **IVTSetup** .
6. Spusťte skript IVTRun zadáním následujícího příkazu:

```
IVTRun -url providerURL [-icf initCtxFact]
```

Obdržíte výstup podobný následujícímu 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, 2024. All Rights Reserved.
Starting WebSphere MQ classes for Java Message Service Administration

InitCtx>
InitCtx>
InitCtx>
InitCtx>
InitCtx>
Stopping MQSeries classes for Java 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, 2024. All Rights Reserved.
MQSeries classes for Java 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 Java pomocí úložiště Maven

Při vývoji aplikace Java pro produkt IBM MQpomocí úložiště Maven k automatické instalaci závislostí nemusíte nic explicitně instalovat před použitím rozhraní IBM MQ .

Centrální úložiště Maven

Maven je nástroj pro sestavování aplikací a také poskytuje úložiště pro ukládání artefaktů, ke kterým může vaše aplikace chtít přistupovat.

Úložiště Maven (nebo centrální úložiště) má strukturu, která umožňuje, aby soubory, jako jsou soubory JAR, měly různé verze, které se pak snadno zjišťují pomocí dobře známého mechanizmu pojmenování. Sestavovací nástroje pak mohou tyto názvy použít k dynamickému stahování závislostí pro vaši aplikaci. V definici vaší aplikace, která se při použití nástroje Maven jako sestavovacího nástroje nazývá soubor POM, pojmenujete závislosti a proces sestavení bude vědět, co odtud dělat.

Soubory klienta IBM MQ

Kopie rozhraní klienta IBM MQ Java jsou k dispozici v centrálním úložišti pod položkou `com.ibm.mq GroupId`. Můžete najít soubor `com.ibm.mq.jakarta.client.jar` (Jakarta Messaging 3.0) a soubor `com.ibm.mq.allclient.jar` (JMS 2.0). Tyto soubory se obvykle používají pro samostatné programy. Můžete také najít soubor `wmq.jakarta.jmsra.rar` (Jakarta Messaging 3.0) a soubor `wmq.jmsra.rar` (JMS 2.0), který se používá na aplikačních serverech Java EE). Hodnoty `jakarta.client.jar` a `allclient.jar` obsahují hodnoty IBM MQ classes for JMS a IBM MQ classes for Java.

Důležité: Použití formátu Apache Maven Assembly Plugin *jar-se-závislostmi* k sestavení aplikace, která obsahuje přemístitelný soubor JAR IBM MQ , není podporováno.

V souboru `pom.xml` zpracovaném příkazem maven přidáte závislosti pro tyto soubory JAR, jak je uvedeno v následujících příkladech:

- V 9.3.0 JM 3.0 V 9.3.0 Chcete-li zobrazit vztah mezi kódem aplikace a produktem `com.ibm.mq.jakarta.client.jar`, postupujte takto:

  ```
  <dependency>
          <groupId>com.ibm.mq</groupId>
          <artifactId>com.ibm.mq.jakarta.client</artifactId>
          <version>9.3.0.0</version>
      </dependency>
  ```

- JMS 2.0 Chcete-li zobrazit vztah mezi kódem aplikace a produktem `com.ibm.mq.allclient.jar`, postupujte takto:

  ```
  <dependency>
          <groupId>com.ibm.mq</groupId>
          <artifactId>com.ibm.mq.allclient</artifactId>
          <version>9.2.2.0</version>
      </dependency>
  ```

- V 9.3.0 JM 3.0 V 9.3.0 Pro použití adaptéru prostředků Jakarta EE :

```
<dependency>
     <groupId>com.ibm.mq</groupId>
     <artifactId>wmq.jakarta.jmsra</artifactId>
     <version>9.3.0.0</version>
 </dependency>
```

- JMS 2.0 Pro použití adaptéru prostředků JMS 2.0 Java EE :

```
<dependency>
     <groupId>com.ibm.mq</groupId>
     <artifactId>wmq.jmsra</artifactId>
     <version>9.2.2.0</version>
 </dependency>
```

Příklad jednoduchého projektu v produktu Eclipse pro spuštění projektu JMS viz IBM Developer článek Vývoj Java aplikací pro produkt MQ , který je nyní snazší s nástrojem Maven.

Vývoj aplikací C++

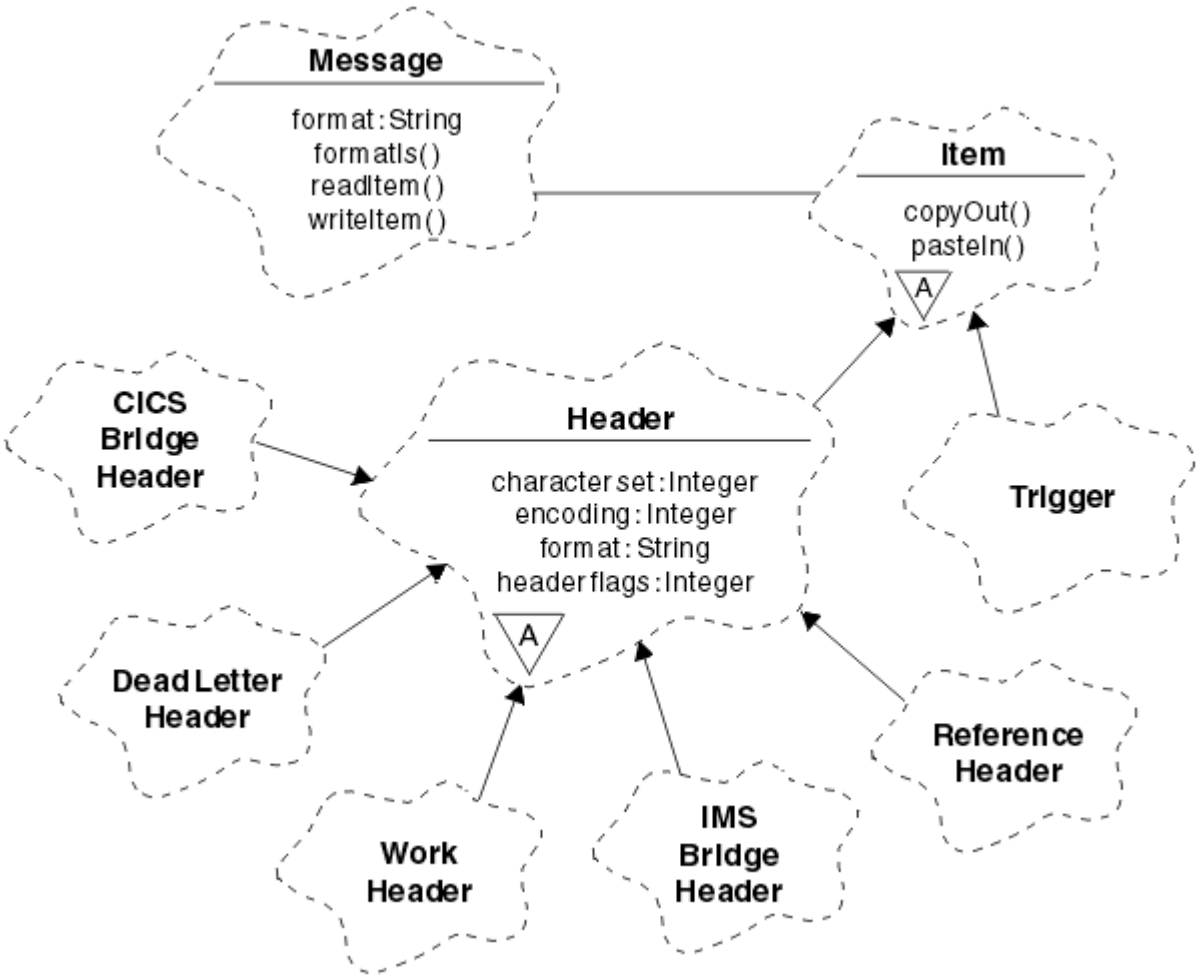
Produkt IBM MQ poskytuje třídy C++ ekvivalentní objektům IBM MQ a některé další třídy ekvivalentní datovým typům pole. Poskytuje řadu funkcí, které nejsou k dispozici prostřednictvím rozhraní MQI.

IBM WebSphere MQ 7.0, rozšíření programovacích rozhraní IBM MQ se nepoužijí na třídy C + +.

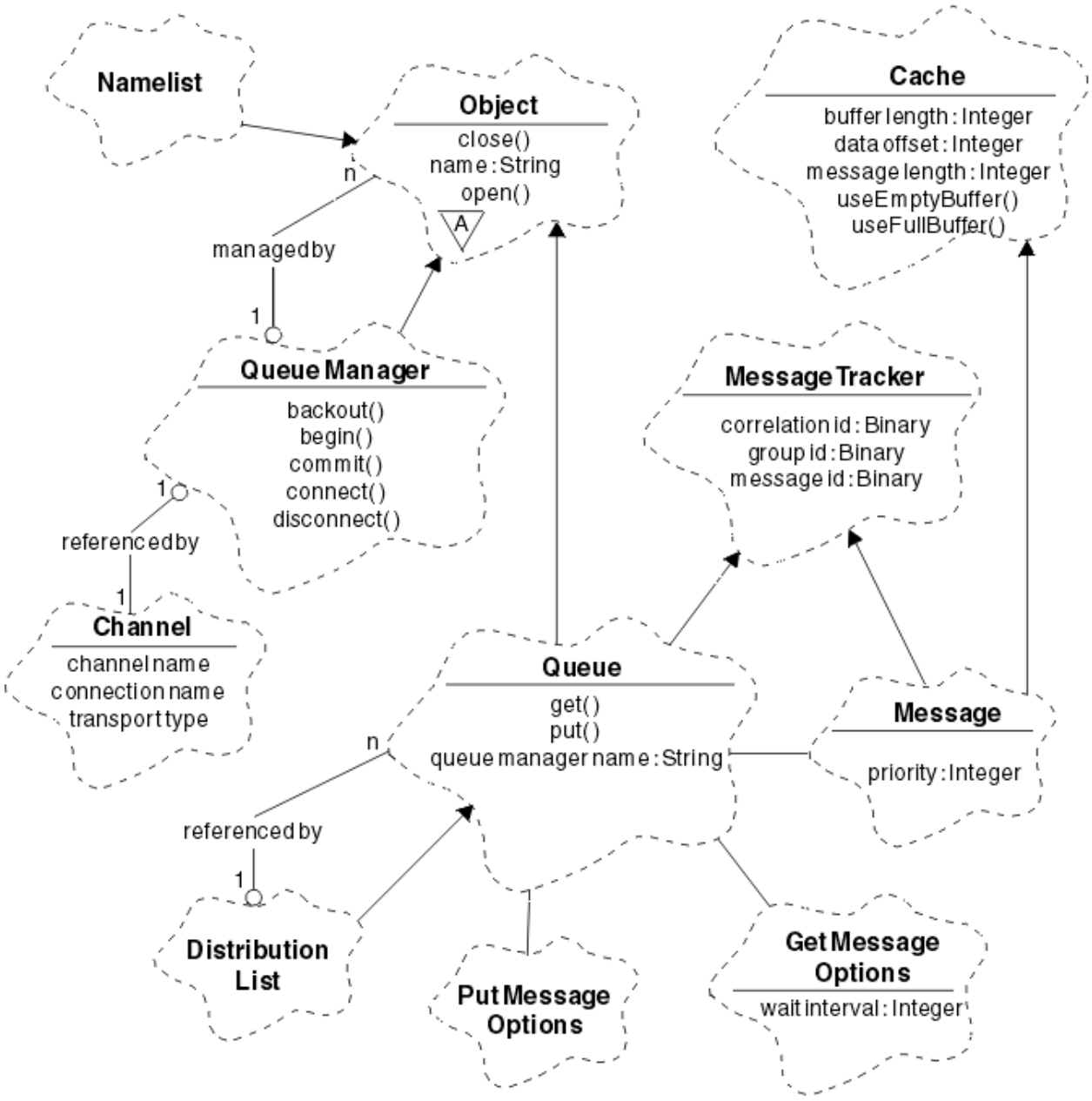
Produkt IBM MQ C++ poskytuje následující funkce:

- Automatická inicializace datových struktur IBM MQ .
- Připojení a otevření fronty správce front ve správném čase.
- Implicitní uzavření fronty a odpojení správce front.
- Přenos a příjem hlavičky nedoručenky.
- Přenos a příjem záhlaví mostu IMS .
- Přenos a příjem záhlaví referenční zprávy.
- Potvrzení zprávy spouštěče.
- CICS bridge přenos záhlaví a příjemka.
- Přenos a příjemka záhlaví práce.
- Definice kanálu klienta.

Následující diagramy tříd Booch ukazují, že všechny třídy jsou široce paralelní s entitami IBM MQ v procedurálním rozhraní MQI (například s použitím jazyka C), které mají buď manipulátory, nebo datové struktury. Všechny třídy dědí od třídy ImqError (viz ImqError C++ class), což umožňuje, aby byl ke každému objektu přidružen chybový stav.



Obrázek 50. IBM MQ C + + (obsluha položek)



Obrázek 51. IBM MQ C + + (správa front)

Chcete-li správně interpretovat diagramy tříd Booch, mějte na paměti následující konvence:

- Metody a pozoruhodné atributy jsou zobrazeny pod názvem *class* .
- Malý trojúhelník v rámci cloudu označuje *abstraktní třídu*.
- *Dědičnost* je označena šipkou k nadřízené třídě.
- Nezdobená čára mezi mraky označuje *kooperativní vztah* mezi třídami.
- Řádek zdobený číslem označuje *referenční vztah* mezi dvěma třídami. Číslo označuje počet objektů, které se mohou současně podílet na konkrétním vztahu.

Následující třídy a datové typy se používají v podpisech metod C++ tříd správy front (viz Obrázek 51 na stránce 514). a třídy obsluhy položek (viz Obrázek 50 na stránce 513):

- Třída ImqBinary (viz ImqBinary C++ class), která zapouzdřuje bajtová pole, jako např. MQBYTE24.
- Datový typ ImqBoolean , který je definován jako **typedef unsigned char ImqBoolean**.
- Třída ImqString (viz ImqString C++ class), která zapouzdřuje znaková pole, jako např. MQCHAR64.

Entity s datovými strukturami jsou subponovány v rámci příslušných tříd objektů. Jednotlivá pole datové struktury (viz křížový odkaz C++ a MQI) je k nim přistupováno pomocí metod.

Entity s manipulátory spadají do hierarchie tříd ImqObject (viz ImqObject C++ class). a poskytují zapouzdřená rozhraní pro rozhraní MQI. Objekty těchto tříd vykazují inteligentní chování, které může snížit počet vyvolání metod požadovaných vzhledem k procedurální MQI. Můžete například vytvořit a vyřadit připojení správce front podle potřeby, nebo můžete otevřít frontu s příslušnými volbami a zavřít ji.

Třída ImqMessage (viz ImqMessage C++ class). zapouzdřuje datovou strukturu MQMD a také funguje jako bod zadržení pro uživatelská data a *položky* (viz “Čtení zpráv v C++” na stránce 524) poskytováním vyrovnávacích pamětí uložených v mezipaměti. Můžete poskytnout vyrovnávací paměti s pevnou délkou pro uživatelská data a použít vyrovnávací paměť mnohokrát. Množství dat přítomných ve vyrovnávací paměti se může lišit od jednoho použití k druhému. Alternativně může systém poskytovat a spravovat vyrovnávací paměť s flexibilní délkou. Důležité aspekty se stávají jak velikost vyrovnávací paměti (množství dostupné pro příjem zpráv), tak skutečně použité množství (buď počet bajtů pro přenos, nebo počet skutečně přijatých bajtů).

Související pojmy

Technický přehled

“Ukázkové programy C++” na stránce 515
K dispozici jsou čtyři ukázkové programy, které demonstrují získávání a vkládání zpráv.

“Aspekty týkající se jazyka C++” na stránce 519
V této kolekci témat jsou podrobně popsány aspekty používání jazyka C++ a konvence, které je třeba vzít v úvahu při psaní aplikačních programů používajících rozhraní MQI (Message Queue Interface).

“Příprava dat zprávy v C++” na stránce 523
Data zprávy jsou připravena ve vyrovnávací paměti, kterou může dodat systém nebo aplikace. Existují výhody pro obě metody. Jsou uvedeny příklady použití vyrovnávací paměti.

“Vyvíjení aplikací pro IBM MQ” na stránce 5
Můžete vyvíjet aplikace pro odesílání a příjem zpráv a pro správu správců front a souvisejících prostředků. Produkt IBM MQ podporuje aplikace napsané v mnoha různých jazycích a rámcích.

Související odkazy

“Sestavení programů IBM MQ C++” na stránce 529
Je uvedena URL podporovaných kompilátorů spolu s příkazy, které se mají použít ke kompilaci, propojení a spuštění programů a ukázek C++ na platformách IBM MQ .

Křížový odkaz C++ a MQI

IBM MQ třídy C++

Ukázkové programy C++

K dispozici jsou čtyři ukázkové programy, které demonstrují získávání a vkládání zpráv.

Ukázkové programy jsou:

- AHOJ SVĚTE (imqwrld.cpp)
- SPUT (imqsput.cpp)
- SGET (imqsget.cpp)
- DPUT (imqdput.cpp)

Ukázkové programy jsou umístěny v adresářích uvedených v části Tabulka 73 na stránce 516.

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

Tabulka 73. Umístění ukázkových programů		
Prostředí	**Adresář obsahující zdroj**	**Adresář obsahující sestavení programy**
AIX	*MQ_INSTALLATION_PATH*/samp	*MQ_INSTALLATION_PATH*/samp/bin/ia
V 9.3.5 AIX	*MQ_INSTALLATION_PATH*/samp	*MQ_INSTALLATION_PATH*/samp/bin/ca (viz poznámka "1" na stránce 516)
IBM i	`/QIBM/ProdData/mqm/samp/`	(viz poznámka "2" na stránce 516)
Linux	*MQ_INSTALLATION_PATH*/samp	Není
Windows	*MQ_INSTALLATION_PATH*\tools\cplus\samples	*MQ_INSTALLATION_PATH*\tools\cplus\ukázky \bin\vn (viz poznámka "3" na stránce 516)
z/OS	thlqual.SCSQCPPS	

Notes:

1. V 9.3.5 AIX Programy sestavené pomocí kompilátoru XLC 17 jsou umístěny ve složce "`ca`", zatímco programy sestavené pomocí kompilátoru XLC 16 jsou umístěny ve složce "`ia`".
2. IBM i Programy sestavené pomocí kompilátoru ILE C++ pro produkt IBM i jsou v knihovně QMQM. Zdrojové soubory jsou v adresáři `/QIBM/ProdData/mqm/samp`.
3. Windows Programy sestavené pomocí produktu Microsoft Visual Studio Visual Studio se nacházejí v adresáři *MQ_INSTALLATION_PATH*`\tools\cplus\samples\bin\vn`. Další informace o těchto kompilátorech viz "Sestavení programů C++ na Windows" na stránce 535.

Ukázkový program HELLO WORLD (imqwrld.cpp)

Tento ukázkový program C + + ukazuje, jak vložit a získat běžný datagram (strukturu C) pomocí třídy ImqMessage .

Tento program ukazuje, jak vložit a získat běžný datagram (struktura C) pomocí třídy ImqMessage . Tato ukázka používá několik vyvolání metod s využitím implicitních vyvolání metod, jako např. **open**, **close**a **disconnect**.

Na všech platformách kromě operačního systému z/OS

Pokud používáte připojení serveru k produktu IBM MQ, postupujte takto:

- Chcete-li použít existující výchozí frontu, SYSTEM.DEFAULT.LOCAL.QUEUE, spusťte program **imqwrlds** bez předání parametrů
- Chcete-li použít dočasnou dynamicky přiřazenou frontu, spusťte příkaz **imqwrlds** s předáním názvu výchozí modelové fronty SYSTEM.DEFAULT.MODEL.QUEUE.

Pokud používáte připojení klienta k produktu IBM MQ, postupujte podle jedné z následujících procedur:

- Nastavte proměnnou prostředí MQSERVER (další informace viz *MQSERVER*) a spusťte příkaz **imqwrldc**, nebo

- Spusťte příkaz **imqwrldc** s předáním parametrů **queue-name**, **queue-manager-name**a **channel-definition**, kde typický parametr **channel-definition** může být SYSTEM.DEF.SVRCONN/TCP/*název hostitele* (1414)

zapz/OS

z/OS

Vytvořte a spusťte dávkovou úlohu pomocí ukázkového JCL **imqwrldr**.

Další informace viz z/OS Dávka, Dávka RRS a CICS .

Vzorový 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 ;
}
```

Ukázkové programy SPUT (imqsput.cpp) a SGET (imqsget.cpp)

Tyto programy C++ umisťují zprávy do pojmenované fronty a načítají z ní zprávy.

Tyto ukázky ukazují použití následujících tříd:

- ImqError (viz ImqError třída C++)
- ImqMessage (viz ImqMessage třída C++)
- ImqObject (viz ImqObject třída C++)
- ImqQueue (viz ImqQueue třída C++)

- ImqQueueManager (viz ImqQueuetřída správce C++)

Při spouštění programů postupujte podle příslušných pokynů.

Na všech platformách kromě operačního systému z/OS

1. Spusťte příkaz **imqsputs** *název_fronty*.
2. Zadejte řádky textu na konzole. Tyto řádky jsou umístěny jako zprávy do uvedené fronty.
3. Zadejte nulový řádek pro ukončení vstupu.
4. Spusťte příkaz **imqsgets** *název-fronty* , abyste načetli všechny řádky a zobrazili je na konzole.

z/OS Další informace viz “Sestavení programů C++ v systémech z/OS Batch, RRS Batch a CICS” na stránce 537.

zapz/OS

z/OS

1. Vytvořte a spusťte dávkovou úlohu pomocí ukázkového JCL **imqsputr**. Zprávy jsou čteny z datové sady SYSIN.
2. Vytvořte a spusťte dávkovou úlohu pomocí ukázkového JCL **imqsgetr**. Zprávy jsou načteny z fronty a odeslány do datové sady SYSPRINT.

Ukázkový program DPUT (imqdput.cpp)

Tento ukázkový program v jazyce C++ vkládá zprávy do rozdělovníku, který se skládá ze dvou front.

DPUT zobrazuje použití třídy ImqDistributionList (viz ImqDistributionList C++ class). Tato ukázka není v systému z/OSpodporována.

1. Spusťte příkaz **imqdputs** *queue-name-1 queue-name-2* , abyste umístili zprávy do dvou pojmenovaných front.
2. Spuštěním příkazu **imqsgets** *queue-name-1* a **imqsgets** *queue-name-2* načtěte zprávy z těchto front.

Aspekty týkající se jazyka C++

V této kolekci témat jsou podrobně popsány aspekty používání jazyka C++ a konvence, které je třeba vzít v úvahu při psaní aplikačních programů používajících rozhraní MQI (Message Queue Interface).

Soubory záhlaví C++

Soubory záhlaví jsou poskytovány jako součást definice MQI, aby vám pomohly psát aplikační programy IBM MQ v jazyce C + +.

Tyto soubory záhlaví jsou shrnuty v následující tabulce.

Tabulka 74. Hlavičkové soubory C/C++

Název souboru	**Obsah**
IMQI.HPP	Třídy C++ MQI (včetně CMQC.H a IMQTYPE.H)
IMQTYPE.H	Definuje datový typ **ImqBoolean** .
CMQC.H	Datové struktury MQI a konstanty manifestu

Chcete-li zlepšit přenositelnost aplikací, kódujte název hlavičkového souboru malými písmeny v direktivě preprocesoru **#include** :

```
#include <imqi.hpp> // C++ classes
```

Metody a atributy C++

Názvy metod jsou ve velkých i malých písmen. Na parametry a návratové hodnoty se vztahují různé aspekty. K atributům se přistupuje pomocí metod set a get podle potřeby.

Parametry metod, které jsou *const* , jsou 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 stanou odpovědností volajícího.

Některé podpisy metod zahrnují položky, které mají výchozí nastavení, není-li uvedeno. Tyto 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 smíšené, začínající malými písmeny. Každé slovo, kromě prvního v názvu metody, začíná velkým písmenem. Zkratky se nepoužívají, pokud není jejich význam široce pochopen. Použité zkratky zahrnují *id* (pro identitu) a *sync* (pro synchronizaci).

K atributům objektu se přistupuje pomocí metod set a get. Metoda set začíná slovem *set* ; metoda get nemá žádnou předponu. Je-li atribut *jen pro čtení*, neexistuje žádná metoda nastavení.

Atributy jsou inicializovány do platných stavů 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 **nepodepsaný znak** v IMQTYPE.H a může mít hodnoty TRUE a FALSE. Namísto polí **MQBYTE** můžete použít objekty třídy **ImqBinary** a namísto objektů třídy **char ***můžete použít objekty třídy **ImqString** . Mnoho metod vrací objekty namísto ukazatelů **char** nebo **MQBYTE** pro usnadnění správy úložiště. Všechny návratové hodnoty se stanou zodpovědností volajícího a v případě vráceného objektu lze úložiště zlikvidovat pomocí odstranění.

Manipulace s binárními řetězci v C++

Řetězce binárních dat jsou deklarovány jako objekty třídy **ImqBinary** . Objekty této třídy lze kopírovat, porovnávat a nastavovat pomocí známých operátorů C. Je uveden vzorový kód.

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 vracena v objektech třídy **ImqString** , které lze přetypovat na **char *** pomocí operátoru převodu. Třída ImqString obsahuje metody, které pomáhají při zpracování znakových řetězců.

Když jsou znaková data přijata nebo vrácena pomocí metod MQI C++, jsou znaková data vždy ukončena s hodnotou null a mohou mít libovolnou délku. Avšak určité limity jsou stanoveny produktem IBM MQ , což může vést ke zkrácení informací. Pro usnadnění správy úložiště jsou znaková data často vracena v objektech třídy **ImqString** . Tyto objekty lze přetypovat na **char *** pomocí poskytnutého operátoru převodu a použít pro účely *jen pro čtení* v mnoha situacích, kdy je požadován **char *** .

Poznámka: Výsledek převodu **char *** z objektu třídy **ImqString** může mít hodnotu null.

Ačkoli funkce jazyka C lze použít na **char ***, existují speciální metody třídy **ImqString** , které jsou vhodnější; **délka operátoru** () je ekvivalentem **strlen** a **storage** () označuje paměť přidělenou pro znaková data.

Počáteční stav objektů v C++

Všechny objekty mají konzistentní počáteční stav, který se odráží v jejich atributech. Počáteční hodnoty jsou definovány v popisech tříd.

Použití jazyka C z C++

Používáte-li funkce jazyka C z programu C + +, zahrňte příslušná záhlaví.

Následující příklad ukazuje `string.h` zahrnutý v programu C + +:

```
extern "C" {
#include <string.h>
}
```

Konvence pro notaci v jazyce 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 takto:

```
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++

Implicitně může dojít k několika operacím, *právě včas*, aby byly splněny nezbytné podmínky 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 ImqQueueManager je automaticky připojen pro libovolnou metodu, která vede k libovolnému volání MQI (viz křížový odkaz C++ a MQI).

Otevřené

Objekt ImqObject se automaticky otevře pro všechny metody, jejichž výsledkem je volání MQGET, MQINQ, MQPUT nebo MQSET. Pomocí metody **openFor** uveďte jednu nebo více relevantních hodnot **open option** .

Znovu otevřít

Objekt ImqObject je automaticky znovu otevřen pro všechny metody, jejichž výsledkem je volání MQGET, MQINQ, MQPUT nebo MQSET, kde je objekt již otevřen, ale existující **volby otevření** nejsou dostatečné pro úspěšné volání MQI. Objekt je dočasně zavřen pomocí dočasné hodnoty **voleb zavření** MQCO_NONE. Pomocí metody **openFor** přidejte relevantní**volba otevření**.

Opětovné otevření může za určitých okolností způsobit problémy:

- Dočasná dynamická fronta je zničena při zavření a nelze ji znovu otevřít.
- K frontě otevřené pro výlučný vstup (buď explicitně, nebo ve výchozím nastavení) mohou ostatní přistupovat v okně příležitosti během uzavření a opětovného otevření.
- Při zavření fronty dojde ke ztrátě pozice kurzoru procházení. Tato situace nebrání uzavření a opětovnému otevření, ale zabraňuje následnému použití kurzoru, dokud se znovu nepoužije MQGMO_BROWSE_FIRST.
- Kontext poslední načtené zprávy je při zavření fronty ztracen.

Dojde-li k některé z těchto okolností nebo ji lze předvídat, vyhněte se opětovnému otevření explicitním nastavením odpovídajících **voleb otevření** před otevřením objektu (buď explicitně, nebo implicitně).

Nastavení **voleb otevření** explicitně pro složité situace zpracování front vede k lepšímu výkonu a vyhne se problémům spojeným s použitím opětovného otevření.

Zavřít

ImqObject se zavře automaticky v libovolném bodě, kde již není životaschopný stav objektu, například pokud je přerušen odkaz na připojení ImqObject nebo pokud je zničen objekt ImqObject .

Odpojit

Správce ImqQueueje automaticky odpojen v libovolném bodě, v němž již připojení není životaschopné, například je-li přerušen odkaz na připojení ImqObject nebo je-li zničen objekt správce ImqQueue.

Binární a znakové řetězce v C++

Třída ImqString zapouzdřuje tradiční datový formát *char ** . Třída ImqBinary zapouzdřuje binární bajtové pole. Některé metody, které nastavují znaková data, mohou data oříznout.

Metody, které nastavují znaky (**char ***) Data vždy převezmou kopii dat, ale některé metody mohou kopii oříznout, protože určité limity jsou stanoveny produktem IBM MQ.

Třída ImqString (viz ImqString třída C++) zapouzdřuje tradiční **char *** a poskytuje podporu pro:

- Porovnání
- Zřetězení
- Kopírování
- Převod celého čísla na text a z textu na celé číslo
- Extrakce tokenu (slova)
- Překlad velkých písmen

Třída ImqBinary (viz ImqBinary třída C++) zapouzdřuje binární bajtová pole libovolné velikosti. Používá se zejména k uchování následujících atributů:

- **token evidence** (MQBYTE32)
- **značka připojení** (MQBYTE128)
- **ID korelace** (MQBYTE24)
- **token zařízení** (MQBYTE8)
- **id skupiny** (MQBYTE24)
- **ID instance** (MQBYTE24)

- **ID zprávy** (MQBYTE24)
- **token zprávy** (MQBYTE16)
- **ID instance transakce** (MQBYTE16)

Kde tyto atributy patří k objektům následujících tříd:

- ImqCICSBridgeHeader (viz ImqCICSBridgeHeader třída C++)
- ImqGetMessageOptions (viz ImqGetMessageOptions třída C++)
- ImqIMSBridgeHeader (viz ImqIMSBridgeHeader třída C++)
- ImqMessageTracker (viz ImqMessageTřída Tracker C++)
- ImqQueueManager (viz ImqQueuetřída správce C++)
- ImqReferenceZáhlaví (viz ImqReferenceTřída záhlaví C++)
- ImqWorkZáhlaví (viz ImqWorkTřída záhlaví C++)

Třída ImqBinary také poskytuje podporu pro porovnání a kopírování.

Nepodporované funkce v C++

Třídy a metody IBM MQ C++ jsou nezávislé na platformě IBM MQ . Mohou proto nabízet některé funkce, které nejsou na některých platformách podporovány.

Pokud se pokusíte použít funkci na platformě, na které není podporována, funkce je zjištěna produktem IBM MQ , ale ne vazbami jazyka C + +. Produkt IBM MQ ohlásí chybu vašemu programu, stejně jako ostatní chyby rozhraní MQI.

Zasílání zpráv v C++

Tato kolekce témat obsahuje podrobnosti o přípravě, čtení a zápisu zpráv v jazyku C + +.

Příprava dat zprávy v C++

Data zprávy jsou připravena ve vyrovnávací paměti, kterou může dodat systém nebo aplikace. Existují výhody pro obě metody. Jsou uvedeny příklady použití vyrovnávací paměti.

Když odešlete zprávu, data zprávy se nejprve připraví ve vyrovnávací paměti spravované objektem ImqCache (viz ImqCache C++ class). Vyrovnávací paměť je přidružena (dědičností) ke každému objektu ImqMessage (viz ImqMessage třída C++): může být dodána aplikací (pomocí metody **useEmptyBuffer** nebo **useFullBuffer**) nebo automaticky systémem. Výhodou aplikace dodávající vyrovnávací paměť zpráv je, že v mnoha případech není nutné žádné kopírování dat, protože aplikace může přímo využívat připravené datové oblasti. Nevýhodou je, že dodávaná vyrovnávací paměť má pevnou délku.

Vyrovnávací paměť lze znovu použít a počet přenesených bajtů lze pokaždé měnit pomocí metody **setMessageLength** před přenosem.

Při automatickém dodání systémem je dostupný počet bajtů spravován systémem a data lze zkopírovat do vyrovnávací paměti zpráv například pomocí metody ImqCache **write** nebo metody ImqMessage **writeItem** . Vyrovnávací paměť zpráv roste podle potřeby. Jak vyrovnávací paměť roste, nedochází ke ztrátě dříve zapsaných dat. Velká nebo vícedílná zpráva může být napsána postupně.

Následující příklady ukazují zjednodušené odesílání zpráv.

1. Použít připravená data v uživatelem dodané vyrovnávací paměti

```
char szBuffer[ ] = "Hello world" ;

msg.useFullBuffer( szBuffer, sizeof( szBuffer ) );
msg.setFormat( MQFMT_STRING );
```

2. Použít připravená data ve vyrovnávací paměti dodané uživatelem, 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 dodané uživatelem

```
char szBuffer[ 12 ];

msg.useEmptyBuffer( szBuffer, sizeof( szBuffer ) );
msg.setFormat( MQFMT_STRING );
msg.write( 12, "Hello world" );
```

4. Kopírovat data do vyrovnávací paměti dodané systémem

```
msg.setFormat( MQFMT_STRING );
msg.write( 12, "Hello world" );
```

5. Kopírovat data do vyrovnávací paměti dodané systémem pomocí objektů (objekty nastavují formát zprávy i 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 je lze číst sekvenčně. Existuje třída, která je ekvivalentní každému typu zprávy. Je uveden vzorový kód.

Při příjmu dat může aplikace nebo systém dodat vhodnou vyrovnávací paměť zpráv. Stejnou vyrovnávací paměť lze použít jak pro vícenásobný přenos, tak pro vícenásobný příjem pro konkrétní objekt ImqMessage . Je-li vyrovnávací paměť zpráv dodávána automaticky, zvětšuje se tak, aby odpovídala libovolné délce přijatých dat. Vyrovnávací paměť zpráv dodaná aplikací však nemusí být dostatečně velká na to, aby zadržela přijatá data. Pak se může vyskytnout buď oříznutí, nebo selhání, v závislosti na volbách použitých pro příjem zprávy.

K příchozím datům lze přistupovat přímo z vyrovnávací paměti zpráv, v takovém případě délka dat označuje celkové množství příchozích dat. Případně lze příchozí data číst sekvenčně z vyrovnávací paměti zpráv. V tomto případě datový ukazatel adresuje další bajt příchozích dat a datový ukazatel a délka dat se aktualizují při každém čtení dat.

Položky jsou části zprávy, všechny v uživatelské oblasti vyrovnávací paměti zpráv, které je třeba zpracovat postupně a odděleně. Kromě běžných uživatelských dat může být položkou záhlaví nedoručeného dopisu nebo zpráva spouštěče. Položky jsou vždy přidruženy k formátům zpráv; formáty zpráv **nejsou** vždy přidruženy k položkám.

Pro každou položku existuje třída objektu, která odpovídá rozeznatelnému formátu zprávy IBM MQ . Existuje jedna pro záhlaví nedoručených zpráv a jedna pro zprávu spouštěče. Pro uživatelská data neexistuje žádná třída objektu. To znamená, že po vyčerpání rozpoznatelných formátů je zpracování zbytku ponecháno na aplikačním programu. Třídy pro uživatelská data lze zapsat specializací třídy ImqItem .

Následující příklad zobrazuje příjemku zprávy, která zohledňuje řadu potenciálních položek, které mohou předcházet uživatelská data, v imaginární situaci. Uživatelská data, která nejsou položkami, jsou definována jako cokoli, co se vyskytne po položkách, které lze identifikovat. Automatická vyrovnávací paměť (výchozí) se používá k uchování 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 příkladu je FMT_USERCLASS konstanta představující název formátu o délce 8 znaků přidružený k objektu třídy `UserClassa` je definována aplikací.

`UserClass` je odvozen od třídy ImqItem (viz ImqItem třída C++) a implementuje virtuální metody **copyOut** a **pasteIn** z této třídy.

Následující dva příklady ukazují kód ze třídy ImqDeadLetterHeader (viz ImqDeadLetterHeader třída C++). První příklad zobrazuje vlastní zapouzdřenou zprávu- *zápis* kódu.

```
// 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- *čtení* kódu.

```
// 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 ;
}
```

Při použití automatické vyrovnávací paměti je vyrovnávací paměť *nestálá*. To znamená, že po každém vyvolání metody **get** mohou být data vyrovnávací paměti uchovávána v jiném fyzickém umístění. Proto při každém odkazování na data vyrovnávací paměti použijte metody **bufferPointer** nebo **dataPointer** pro přístup k datům zprávy.

Můžete chtít, aby program zrušil pevnou oblast pro příjem dat zprávy. V tomto případě vyvolejte metodu **useEmptyBuffer** před použitím metody **get** .

Použití pevné, neautomatické oblasti omezuje zprávy na maximální velikost, takže je důležité zvážit volbu MQGMO_ACCEPT_TRUNCATED_MSG objektu ImqGetMessageOptions . Není-li tato volba uvedena (předvolba), 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.

Další příklad ukazuje, jak lze pevnou oblast úložiště použít 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 obecný přístup. Aplikace (ne objekt třídy ImqCache) musí vyřadit uživatelem definovanou (neautomatickou) vyrovnávací paměť.

Upozornění: Zadání ukazatele Null a nulové délky pomocí **useEmptyBuffer** nenavržení vyrovnávací paměti s pevnou délkou o nulové délce, jak by mohlo být očekáváno. Tato kombinace je interpretována jako požadavek na ignorování jakékoli předchozí vyrovnávací paměti definované uživatelem a místo toho se vrátí k použití automatické vyrovnávací paměti.

Zápis zprávy do fronty nedoručených zpráv v C++

Vzorový kód programu pro zápis zprávy do fronty nedoručených zpráv.

Typickým případem vícedílné zprávy je zpráva obsahující záhlaví nedoručených zpráv. Data ze zprávy, kterou nelze zpracovat, jsou připojena k záhlaví nedoručeného dopisu.

```
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 do mostu IMS .

Zprávy odeslané do mostu IBM MQ - IMS mohou používat speciální záhlaví. Záhlaví mostu IMS má předponu pro běžná data 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 souboru CICS bridge v C++

Vzorový kód programu pro zápis zprávy do souboru CICS bridge.

Zprávy odeslané do produktu IBM MQ for z/OS pomocí konzoly CICS bridge vyžadují speciální záhlaví. Záhlaví CICS bridge má předponu v datech běžných zpráv.

```
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 );
```

Zápis zprávy s pracovním záhlavím v C++

Vzorový kód programu pro zápis zprávy určené pro frontu spravovanou správcem zátěže z/OS .

Zprávy odeslané do produktu IBM MQ for z/OS, které jsou určeny pro frontu spravovanou správcem pracovní zátěže z/OS , vyžadují speciální záhlaví. Před záhlaví práce jsou uvedena běžná data 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 );
```

Sestavení programů IBM MQ C++

Je uvedena URL podporovaných kompilátorů spolu s příkazy, které se mají použít ke kompilaci, propojení a spuštění programů a ukázek C++ na platformách IBM MQ .

Seznam kompilátorů pro každou podporovanou platformu a verzi produktu IBM MQviz Systémové požadavky pro IBM MQ.

Příkaz, který potřebujete ke kompilaci a propojení programu IBM MQ C++, závisí na vaší instalaci a požadavcích. Následující příklady ukazují typické kompilující a propojující příkazy pro některé kompilátory používající výchozí instalaci produktu IBM MQ na řadě platforem.

AIX Sestavení programů C++ na AIX

Sestavte programy IBM MQ C++ v systému AIX pomocí kompilátoru XL C Enterprise Edition .

V 9.3.5 Další informace o různém mapování voleb kompilátoru mezi kompilátory XLC 16 a XLC 17 naleznete v tématu Mapování voleb.

Deprecated V 9.3.5 Podpora XL C/C++ pro kompilátor AIX 16 na systému AIX je zamítnuta z IBM MQ 9.3.5.

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 -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -limqc23ia -limqb23ia -lmqic
```

32bitová aplikace se závitem

```
xlC_r -o imqsputc_32_r imqsput.cpp -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 -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -limqc23ia -limqb23ia -lmqic
```

64bitová aplikace s podporou podprocesů

```
xlC_r -q64 -o imqsputc_64_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -limqc23ia_r -limqb23ia_r -lmqic_r
```

V 9.3.5 **32bitová aplikace bez podprocesů (XLC 17)**

```
ibm-clang++_r -o imqsputc_32 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -limqc23ca -limqb23ca -lmqic
```

V 9.3.5 **32bitová aplikace se závitem (XLC 17)**

```
ibm-clang++_r -o imqsputc_32_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -limqc23ca_r -limqb23ca_r -lmqic_r
```

V 9.3.5 **64bitová aplikace bez podprocesů (XLC 17)**

```
ibm-clang++_r -m64 -o imqsputc_64 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -limqc23ca -limqb23ca -lmqic
```

V 9.3.5 **64bitová aplikace se závitem (XLC 17)**

```
ibm-clang++_r -m64 -o imqsputc_64_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -limqc23ca_r -limqb23ca_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 -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -limqs23ia -limqb23ia -lmqm
```

32bitová aplikace se závitem

```
xlC_r -o imqsput_32_r imqsput.cpp -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 -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -limqs23ia -limqb23ia -lmqm
```

64bitová aplikace s podporou podprocesů

```
xlC_r -q64 -o imqsput_64_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -limqs23ia_r -limqb23ia_r -lmqm_r
```

V 9.3.5 **32bitová aplikace bez podprocesů (XLC 17)**

```
ibm-clang++_r -o imqsput_32 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -limqs23ca -limqb23ca -lmqm
```

V 9.3.5 **32bitová aplikace se závitem (XLC 17)**

```
ibm-clang++_r -o imqsput_32_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -limqs23ca_r -limqb23ca_r -lmqm_r
```

V 9.3.5 **64bitová aplikace bez podprocesů (XLC 17)**

```
ibm-clang++_r -m64 -o imqsput_64 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -limqs23ca -limqb23ca -lmqm
```

V 9.3.5 **64bitová aplikace se závitem (XLC 17)**

```
ibm-clang++_r -m64 -o imqsput_64_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -limqs23ca_r -limqb23ca_r -lmqm_r
```

IBM i Sestavení programů C++ na IBM i

Sestavte programy IBM MQ C++ na systému IBM i pomocí kompilátoru ILE C + +.

IBM ILE C++ for IBM i je nativní kompilátor pro programy v jazyce C + +. Následující pokyny popisují, jak použít tento kompilátor k vytvoření aplikací IBM MQ C++ pomocí *Ahoj světe!* Ukázkový program IBM MQ jako příklad.

1. Nainstalujte kompilátor ILE C++ for IBM i podle pokynů v části *Nejprve si mě přečtěte!* návod, který doprovází výrobek.
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 C++ lze nalézt v adresáři `/QIBM/ProdData/mqm/samp` a soubory začlenění v adresáři `/QIBM/ProdData/mqm/inc`.

   Alternativně lze zdroj nalézt v knihovně `SRCFILE(QCPPSRC/LIB) SRCMBR(IMQWRLD)`.

b. Svažte s servisními programy dodávanými s produktem IBM MQ, abyste vytvořili programový objekt:

```
CRTPGM PGM(MYLIB/IMQWRLD) MODULE(MYLIB/IMQWRLD) +
BNDSRVPGM(QMQM/IMQB23I4 QMQM/IMQS23I4)
```

Chcete-li sestavit aplikaci s podporou podprocesů, použijte programy služeb opětovného vstupu:

```
CRTPGM PGM(MYLIB/IMQWRLD) MODULE(MYLIB/IMQWRLD) +
BNDSRVPGM(QMQM/IMQB23I4[_R] QMQM/IMQS23I4[_R])
```

c. Spusťte ukázkový program HELLO WORLD pomocí SYSTEM.DEFAULT.LOCAL.QUEUE:

```
CALL PGM(MYLIB/IMQWRLD)
```

Linux Sestavení programů C++ na Linux

Sestavte IBM MQ C++ programy na systému 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
```

32bitová aplikace se závitem

```
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
```

64bitová aplikace s podporou podprocesů

```
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
```

32bitová aplikace se závitem

```
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
```

64bitová aplikace s podporou podprocesů

```
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
```

32bitová aplikace se závitem

```
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
```

64bitová aplikace s podporou podprocesů

```
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
```

32bitová aplikace se závitem

```
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
```

64bitová aplikace s podporou podprocesů

```
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
```

x86-64 (32bitové)

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

Klient: x86-64 (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
```

32bitová aplikace se závitem

```
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
```

64bitová aplikace s podporou podprocesů

```
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: x86-64 (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
```

32bitová aplikace se závitem

```
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
```

64bitová aplikace s podporou podprocesů

```
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
```

Windows Sestavení programů C++ na Windows

Sestavte programy IBM MQ C++ v systému Windows pomocí kompilátoru Microsoft Visual Studio C + +.

Upozornění: Knihovny dodávané produktem IBM MQ jsou dynamické knihovny a nikoli statické knihovny. Produkt IBM MQ poskytuje název "`import libraries`", který lze použít pouze během doby kompilace. Pro běhové prostředí musíte použít dynamické knihovny.

Z produktu IBM MQ 8.0.0 Fix Pack 4produkt IBM MQ dodává redistribuovatelné klienty, které obsahují knihovny nezbytné pro spuštění aplikací IBM MQ . Tyto knihovny lze zabalit a znovu distribuovat s klientskými aplikacemi. Další informace viz Redistributable clients on Windows.

Soubory knihovny (.lib) a soubory knihovny 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
```

Instalace univerzálního běhového prostředí C

Používáte-li produkt Windows 8.1 nebo Windows Server 2012 R2, musíte nainstalovat univerzální aktualizaci běhového prostředí C (Universal CRT) z Microsoft. Toto běhové prostředí je součástí produktu Windows 10 a Windows Server 2016.

Univerzální aktualizace CRT je Microsoft update KB3118401. Můžete zkontrolovat, zda máte tuto aktualizaci, vyhledáním souboru s názvem `ucrtbase.dll` ve vašem adresáři `C:\Windows\System32` . Pokud ne, můžete aktualizaci stáhnout z následující Microsoft stránky: https://www.catalog.update.microsoft.com/Search.aspx?q=kb3118401.

Při pokusu o spuštění programu IBM MQ nebo programu, který jste sami zkompilovali pomocí produktu Microsoft Visual Studio 2017, bez nainstalovaného běhového prostředí dojde k chybám, například k následující chybě:

```
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.
```

Poskytování běhových prostředí pro programy Microsoft Visual Studio 2012

Pokud jste zkompilovali program IBM MQ pomocí Microsoft Visual Studio 2012, mějte na paměti, že instalační program IBM MQ neinstaluje běhová prostředí Microsoft Visual Studio 2012 C/C + +. Pokud byla vaše předchozí verze produktu IBM MQ nainstalována na stejném počítači, běhová prostředí produktu Microsoft Visual Studio 2012 jsou k dispozici z této instalace.

Pokud však používáte program, který byl sestaven pomocí produktu Microsoft Visual Studio 2012 , a nebyla nainstalována žádná předchozí verze produktu IBM MQ , musíte provést jednu z následujících akcí:

- Stáhněte a nainstalujte soubor **Microsoft Visual C++ Redistributable for VisualStudio 2017 (32 and 64-bit versions)** z adresáře Microsoft.
- Znovu zkompilujte program s úrovní Microsoft Visual Studio 2017nebo jinou úrovní systému Microsoft Visual Studio , pro kterou jsou běhová prostředí nainstalována.

Knihovny klienta C++ vytvořené pomocí kompilátoru Microsoft Visual Studio 2015

Produkt IBM MQ poskytuje knihovny klienta C++, které jsou sestaveny pomocí kompilátoru Microsoft Visual Studio 2015 C++ a kompilátoru Microsoft Visual Studio 2017 C + +.

K dispozici jsou 32bitové i 64bitové verze knihoven IBM MQ C + +. 32bitové knihovny jsou nainstalovány ve složce `bin\vs2015` a 64bitové knihovny jsou nainstalovány ve složkách `bin64\vs2015` .

Standardně je produkt IBM MQ nakonfigurován tak, aby používal knihovny Microsoft Visual Studio 2017 . Chcete-li použít knihovny Microsoft Visual Studio 2015 , musíte nastavit proměnnou prostředí `MQ_PREFIX_VS_LIBRARIES` na hodnotu `MQ_PREFIX_VS_LIBRARIES=vs2015` před instalací produktu IBM MQnebo před použitím příkazu **setmqenv** či **setmqinst** .

Použití jinak pojmenovaných knihoven IBM MQ C++

Produkt IBM MQ poskytuje některé další knihovny klienta C++, které jsou pojmenovány odlišně. Tyto knihovny jsou sestaveny pomocí kompilátorů Microsoft Visual Studio 2015 a Microsoft Visual Studio 2017 C++. Tyto knihovny jsou poskytovány kromě existujících knihoven C + +, které jsou také sestaveny pomocí kompilátoru Microsoft Visual Studio 2017 C + +. Vzhledem k tomu, že tyto další knihovny IBM MQ C++ mají různé názvy, můžete spustit aplikace IBM MQ C++, které jsou sestaveny pomocí jazyka IBM MQ C++ a kompilovány pomocí produktu Microsoft Visual Studio 2017 a předchozích verzí produktu na stejném počítači.

Další knihovny Microsoft Visual Studio 2017 mají následující názvy:

- `imqb23vnvs2017.dll`
- `imqc23vnvs2017.dll`
- `imqs23vnvs2017.dll`
- `imqx23vnvs2017.dll`

Další knihovny Microsoft Visual Studio 2015 mají následující názvy:

- `imqb23vnvs2015.dll`
- `imqc23vnvs2015.dll`
- `imqs23vnvs2015.dll`
- `imqx23vnvs2015.dll`

K dispozici jsou 32bitové i 64bitové verze těchto knihoven. 32bitové knihovny jsou nainstalovány ve složce `bin` a 64bitové knihovny jsou nainstalovány ve složce `bin64` . Odpovídající knihovny importu jsou nainstalovány v adresářích `Tools\lib` a `Tools\lib64` .

Pokud vaše aplikace používá soubory `imq*vs2015.lib` , musíte ji zkompilovat pomocí kompilátoru Microsoft Visual Studio 2015 . Chcete-li spustit aplikace IBM MQ C++, které jsou kompilovány s produktem Microsoft Visual Studio 2015, nebo aplikace, které jsou kompilovány s dřívější verzí produktu na stejném počítači, musí mít proměnná prostředí PATH předponu, jak je uvedeno v následujících příkladech:

- Pro 32bitové aplikace:

```
SET PATH=installation folder\bin\vs2015;%PATH%
```

- Pro 64bitové aplikace:

```
SET PATH=installation folder\bin64\vs2015;%PATH%
```

Související pojmy

Windows: změny z IBM MQ 8.0

z/OS Sestavení programů C++ v systémech z/OS Batch, RRS Batch a CICS

Sestavte programy IBM MQ C++ v systému z/OS pro prostředí Batch, RRS nebo CICS a spusťte ukázkové programy.

Programy C++ můžete psát pro tři prostředí, která produkt IBM MQ for z/OS podporuje:

- Dávka
- Dávka RRS
- CICS

Kompilovat, předběžně propojit a propojit

Vytvořte aplikaci z/OS kompilací, předběžným propojením a úpravou odkazů vašeho zdrojového kódu C + +.

IBM MQ C++ for z/OS je implementováno jako z/OS DLL pro jazyk IBM C++ for z/OS . Pomocí DLL zřetězíte dodané definice sidedecks s výstupem kompilátoru v době před propojením. To umožňuje spojovacím programu zkontrolovat volání členských funkcí IBM MQ C + +.

Poznámka: Pro každé ze tří prostředí jsou k dispozici tři sady vleček.

Chcete-li sestavit aplikaci IBM MQ for z/OS C++, vytvořte a spusťte JCL. Postupujte takto:

1. Pokud je vaše aplikace spuštěna v adresáři CICS, použijte k překladu příkazů CICS ve vašem programu proceduru dodávanou s produktem CICS.

   V případě aplikací CICS je navíc třeba provést následující akce:

   a. Přidejte knihovnu SCSQLOAD do zřetězení DFHRPL.

   b. Definujte skupinu CSQCAT1 CEDA pomocí člena IMQ4B100 v knihovně SCSQPROC.

   c. Nainstalujte produkt CSQCAT1.
2. Kompilujte program, abyste vytvořili kód objektu. JCL pro vaši kompilaci musí obsahovat příkazy, které zpřístupní soubory definice dat produktu kompilátoru. Definice dat jsou dodávány v následujících knihovnách IBM MQ for z/OS :

   - **thlqual**.SCSQC370
   - **thlqual**.SCSQHPPS

Ujistěte se, že jste zadali volbu kompilátoru `/cxx` .

Poznámka: Název **thlqual** je kvalifikátor vyšší úrovně IBM MQ instalační knihovny na systému z/OS.

3. Předběžně propojte kód objektu vytvořený v kroku “2” na stránce 537, včetně následujících sidedecks definic, které jsou dodávány v **thlqual**.SCSQDEFS:
   a. imqs23dm a imqb23dm pro dávku
   b. 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 538, vytvořte zaváděcí modul a uložte jej v zaváděcí knihovně aplikace.

Chcete-li spustit dávkové programy nebo dávkové programy RRS, zahrňte knihovny **thlqual**.SCSQAUTH a **thlqual**.SCSQLOAD do zřetězení datové sady STEPLIB nebo JOBLIB.

Chcete-li spustit program CICS , nejprve požádejte administrátora systému, aby jej definoval pro CICS jako program a transakci IBM MQ . Pak ji můžete spustit obvyklým způsobem.

Spustit ukázkové programy

Programy jsou popsány v části “Ukázkové programy C++” na stránce 515.

Ukázkové aplikace jsou dodávány pouze ve zdrojové podobě. Jedná se o tyto soubory:

Tabulka 75. Soubory ukázkového programu z/OS

Ukázka	Zdrojový program (v knihovně thlqual.SCSQCPPS)	JCL (v knihovně thlqual.SCSQPROC)
Hello World	imqwrld, nevěstí	imqwrldr
SPUT (vložení)	imqsput	imqsputr
SGET (změna)	imqsget (získání)	imqsgetr

Chcete-li ukázky spustit, zkompilujte je a propojte je s libovolným programem v jazyce C + + (viz “Sestavení programů C++ v systémech z/OS Batch, RRS Batch a CICS” na stránce 537). Pomocí dodaného JCL sestavte a spusťte dávkovou úlohu. Skript JCL musíte nejprve upravit podle komentáře, který je s ním obsažen.

z/OS Sestavení programů C++ na z/OS UNIX System Services

Sestavte programy IBM MQ C++ v systému z/OS UNIX System Services (z/OS UNIX).

Chcete-li sestavit aplikaci v shellu z/OS UNIX , musíte poskytnout kompilátoru přístup k souborům IBM MQ include (umístěným v `thlqual.SCSQC370` a `hlqual.SCSQHPPS`) a odkaz na dva z DLL sidedecks (umístěných v `thlqual.SCSQDEFS`). Za běhu aplikace potřebuje přístup k IBM MQ datovým sadám `thlqual.SCSQLOAD`, `thlqual.SCSQAUTH`a jedné z datových sad specifických pro jazyk, například `thlqual.SCSQANLE` .[6].

[6] Můžete vytvořit propojení s kterýmkoli z vlečných bodů uvedených v části " Předem propojit kód objektu , abyste spustili produkt z/OS UNIX v kterémkoli ze tří prostředí, “Sestavení programů C++ v systémech z/OS Batch, RRS Batch a CICS” na stránce 537

Kompilace

1. Zkopírujte ukázku do systému souborů pomocí příkazu TSO **oput** nebo použijte FTP. Zbytek tohoto příkladu předpokládá, že jste zkopírovali ukázku do adresáře s názvem `/u/fred/samplea` pojmenovali ji `imqwrld.cpp`.
2. Přihlaste se do shellu z/OS UNIX a přejděte do adresáře, kam jste umístili ukázku.
3. Nastavte kompilátor C++ tak, aby mohl přijmout soubory DLL sidedeck a .cpp jako vstup:

```
/u/fred/sample:> export _CXX_EXTRA_ARGS=1
/u/fred/sample:> export _CXX_CXXSUFFIX="cpp"
```

4. Zkompilujte a propojte ukázkový program. Následující příkaz propojí program s dávkovými sidedecks; místo toho lze použít dávkové sidedecks 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 Command Reference.

Můžete také použít obslužný program make pro zjednodušení sestavení programů C + +. Zde je ukázka souboru Makefile pro sestavení ukázkového programu HELLO WORLD C + +. Odděluje fáze sestavování a propojování. Před spuštěním příkazu make nastavte prostředí jako v kroku “3” na stránce 539 .

```
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 příkazu make naleznete v tématu z/OS UNIX System Services Programovací nástroje .

Spuštěno

1. Přihlaste se do shellu z/OS UNIX a přejděte do adresáře, kde jste sestavili ukázku.
2. Nastavte proměnnou prostředí STEPLIB tak, aby zahrnovala datové sady 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 .NET

IBM MQ classes for .NET Povolit aplikacím .NET připojit se k produktu IBM MQ jako k serveru IBM MQ MQI client nebo se připojit přímo k serveru IBM MQ .

Máte-li aplikace, které používají produkt Microsoft .NET Framework a chcete-li využít zařízení produktu IBM MQ, musíte použít IBM MQ classes for .NET. Další informace viz “Instalace produktu IBM MQ classes for .NET Framework” na stránce 547.

Produkt IBM MQ 9.1.1 IBM MQ podporuje .NET Core pro aplikace v prostředí Windows . Další informace viz “Instalace produktu IBM MQ classes for .NET” na stránce 541.

Produkt IBM MQ 9.1.2 IBM MQ podporuje .NET Core pro aplikace v prostředí Linux .

V produktu IBM MQ 9.1.4jsou spravované aplikace produktu IBM MQ .NET schopny automaticky vyvážit připojení v rámci klastrovaných správců front. Podporovány jsou knihovny .NET Framework i .NET Standard . Další informace naleznete v tématu O uniformních klastrech a Automatické vyrovnávání aplikací.

Objektově orientované rozhraní produktu IBM MQ .NET se liší od rozhraní MQI v tom, že používá spíše metody objektů než použití příkazových slov MQI.

Procedurální rozhraní API IBM MQ je sestaveno na základě sloves, jako jsou ta v následujícím seznamu:

```
MQCONN, MQDISC, MQOPEN, MQCLOSE,
MQINQ, MQSET, MQGET, MQPUT, MQSUB
```

Všechna tato příkazová slova přijímají jako parametr popisovač objektu IBM MQ , se kterým mají pracovat. Vzhledem k tomu, že .NET je objektově orientované, programovací rozhraní .NET toto zaokrouhlí. Váš program se skládá ze sady objektů IBM MQ , které se používají při volání metod pro tyto objekty. Programy můžete psát v libovolném jazyce podporovaném produktem .NET.

Použijete-li procedurální rozhraní, odpojíte se od správce front pomocí volání MQDISC (*Hconn*, CompCode, Reason), kde *Hconn* je manipulátor pro správce front.

V rozhraní .NET je správce front reprezentován objektem třídy MQQueueManager. Od správce front se odpojíte voláním metody Disconnect () pro danou třídu.

```
// 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 .NET interakci s produktem IBM MQ. Představují různé komponenty produktu IBM MQ , které vaše aplikace používá, například správce front, fronty, kanály a zprávy. Podrobnosti o těchto třídách viz Třídy a rozhraní IBM MQ .NET.

Před kompilací všech aplikací, které píšete, musíte mít nainstalovaný rámec .NET Framework. Pokyny k instalaci produktu IBM MQ classes for .NET a rámce .NET Framework viz “Instalace produktu IBM MQ classes for .NET Framework” na stránce 547.

Související pojmy

Technický přehled

“Vyvíjení aplikací pro IBM MQ” na stránce 5
Můžete vyvíjet aplikace pro odesílání a příjem zpráv a pro správu správců front a souvisejících prostředků. Produkt IBM MQ podporuje aplikace napsané v mnoha různých jazycích a rámcích.

Související úlohy

Kontaktování podpory společnosti IBM

Odstraňování problémů s produktem IBM MQ.NET

“Vývoj aplikací Microsoft Windows Communication Foundation pomocí IBM MQ” na stránce 1222
Vlastní kanál produktu Microsoft Windows Communication Foundation (WCF) pro produkt IBM MQ odesílá a přijímá zprávy mezi klienty a službami WCF.

“Vývoj aplikací XMS .NET” na stránce 598
IBM MQ Message Service Client (XMS) for .NET (XMS .NET) poskytuje rozhraní API s názvem XMS , které má stejnou sadu rozhraní jako Java Message Service (JMS) Rozhraní API. IBM MQ Message Service Client (XMS) for .NET obsahuje plně spravovanou implementaci XMS, kterou může používat libovolný .NET vyhovující jazyk.

Windows Linux Instalace produktu IBM MQ classes for .NET

Produkt IBM MQ classes for .NET, včetně ukázek, se instaluje s operačním systému IBM MQ na systémech Windows a Linux .

Předpoklady a instalace

V systému IBM MQ 9.2.0je klientská knihovna produktu IBM MQ .NET sestavená pomocí .NET Standard k dispozici v systémech Windows a Linux. Chcete-li spustit produkt IBM MQ classes for .NET Standard, musíte nainstalovat produkt Microsoft .NET Core. Microsoft.NET Core 3.1 je minimální požadovaná verze pro spuštění produktu IBM MQ classes for .NET Standard.

V 9.3.0 V 9.3.0 V systému IBM MQ 9.3.0 IBM MQ podporuje .NET 6 aplikace používající IBM MQ classes for .NET Standard. Používáte-li aplikaci .NET Core 3.1 , můžete spustit tuto aplikaci s malou úpravou v souboru `csproj` , nastavením `targetframeworkversion` na hodnotu `"net6.0"`, bez nutnosti rekompilace.

V 9.3.1 Produkt IBM MQ 9.3.1 poskytuje knihovnu klienta IBM MQ .NET sestavenou pro .NET 6 jako cílový rámec. V produktu IBM MQ 9.3.1je Microsoft .NET 6.0 minimální požadovaná verze pro spouštění aplikací pomocí knihoven IBM MQ , které jsou sestaveny pomocí .NET 6 jako cílový rámec.

V 9.3.1 V systému IBM MQ 9.3.1je knihovna klienta IBM MQ .NET sestavená pomocí .NET Standard k dispozici v nové složce `netstandard2.0` a knihovna klienta IBM MQ .NET sestavená pomocí .NET 6 jako cílový rámec je k dispozici v systému *MQ_INSTALLATION_PATH*`/bin` v systému Windows a v systému *MQ_INSTALLATION_PATH*`/lib64` v systému Linux.

Nejnovější verze produktu IBM MQ classes for .NET se standardně instaluje jako součást standardní instalace produktu IBM MQ ve funkci *Java a .NET Messaging a Web Services* .

Windows Další informace o předpokladech a instalaci na webu Windows:

- Viz Požadavky na produkt IBM MQ classes for .NET, chcete-li předem vyžadovaný software spustit IBM MQ classes for .NET.
- Pokyny k instalaci viz Instalace serveru IBM MQ na Windows nebo Instalace klienta IBM MQ na Windows systémech .

Linux Další informace o předpokladech a instalaci na webu Linux:

- Viz Požadavky na produkt IBM MQ classes for .NET, chcete-li předem vyžadovaný software spustit IBM MQ classes for .NET.
- Pokyny k instalaci rpm naleznete v tématu Instalace klienta IBM MQ na systémech Linux.
- V případě systému Linux Ubuntu, který používá balíky Debian , se podívejte na téma Instalace klienta IBM MQ na systémech Linux.

Knihovna IBM MQ classes for .NET Standard , `amqmdnetstd.dll`, je k dispozici pro stažení z úložiště NuGet . Další informace viz “Stažení IBM MQ classes for .NET z úložiště NuGet” na stránce 546.

`amqmdnetstd.dll` knihovna

V 9.3.1 V produktu IBM MQ 9.3.1je knihovna `amqmdnetstd.dll` k dispozici v následujících umístěních.

Knihovna sestavená pomocí .NET Standard 2.0 jako cílového rámce

- Windows V systému Windows: *MQ_INSTALLATION_PATH*`\bin\netstandard2.0`.
- Linux V systému Linux: *MQ_INSTALLATION_PATH*`\lib64\netstandard2.0`.

Deprecated Tyto knihovny jsou zamítnuté a produkt IBM je hodlá v budoucích verzích odebrat.

Knihovna sestavená pomocí .NET 6 jako cílového rámce

- Windows V systému Windows: *MQ_INSTALLATION_PATH*\bin. Ukázkové aplikace jsou nainstalovány v produktu *MQ_INSTALLATION_PATH*/samp/dotnet/samples/cs/core/base.
- Linux V systému Linux: *MQ_INSTALLATION_PATH*\lib64. Ukázky .NET jsou v adresáři *MQ_INSTALLATION_PATH*/samp/dotnet/samples/cs/core/base.

LTS Pro produkt IBM MQ 9.3.0 Long Term Supportje knihovna amqmdnetstd.dll k dispozici v následujících umístěních:

- Windows V systému Windows: *MQ_INSTALLATION_PATH*\bin. Ukázkové aplikace jsou nainstalovány v produktu *MQ_INSTALLATION_PATH*/samp/dotnet/samples/cs/core/base.
- Linux V systému Linux: *MQ_INSTALLATION_PATH*/lib64 path. Ukázky .NET jsou v adresáři *MQ_INSTALLATION_PATH*/samp/dotnet/samples/cs/core/base.

Upozornění: Deprecated V 9.3.1 Z IBM MQ 9.3.1, IBM MQ .NET klientské knihovny sestavené pomocí .NET Standard 2.0 jako cílový rámec jsou zamítnuty a aplikace odkazující na tyto knihovny vygenerují během doby kompilace varování CS0618 .

Stabilized LTS Knihovna amqmdnet.dll pro .NET Framework je stále dodána, ale tato knihovna je stabilizována; to znamená, že do ní nebudou zavedeny žádné nové funkce. Pro všechny nejnovější funkce musíte provést migraci do knihovny amqmdnetstd.dll . Můžete však pokračovat v používání knihovny amqmdnet.dll na IBM MQ 9.1 nebo novějších Long Term Support nebo Continuous Delivery vydáních.

V 9.3.1 Pokud je aplikace .NET Framework kompilována pomocí amqmdnetstd.dll nebo amqmxmsstd.dll z verze nižší než IBM MQ 9.3.1 a stejná aplikace je spuštěna pomocí knihoven klienta .NET 6 založených na IBM MQ , pak .NETvygeneruje výjimku typu FileLoadException :

```
Byla zachycena výjimka: System.IO.FileLoadException: Nelze načíst soubor nebo sestavení
'amqmdnetstd, Version =x.x.x.x, Culture=neutral, PublicKeyToken=23d6cb914eeaac0e' nebo
jedna z jeho závislostí. Definice souboru typu manifest umístěného sestavení neodpovídá
odkaz na sestavení. (Výjimka z HRESULT: 0x80131040)

Název souboru: ' amqmdnetstd, Version =x.x.x.x, Culture=neutral,
PublicKeyToken=23d6cb914eeaac0e'
```

Chcete-li vyřešit tuto chybu, knihovny přítomné v produktu *MQ_INSTALLATION_PATH*/bin/netstandard2.0 musí být zkopírovány do adresáře, ze kterého je spuštěna aplikace .NET Framework .

dspmqver příkaz

Pomocí příkazu **dspmqver** můžete zobrazit informace o verzi a sestavení pro komponentu .NET Core .

Porovnání funkcí mezi knihovnami IBM MQ classes for .NET Framework a IBM MQ classes for .NET (.NET Standard a .NET 6)

Následující tabulka uvádí funkce pro produkt IBM MQ classes for .NET Framework v porovnání s funkcemi pro knihovny IBM MQ classes for .NET (.NET Standard a .NET 6).

Tabulka 76. Rozdíly mezi knihovnami IBM MQ classes for .NET Framework a IBM MQ classes for .NET (.NET Standard a .NET 6)

Funkce	IBM MQ classes for .NET Framework	IBM MQ classes for .NET (knihovny.NET Standard a .NET 6)
Názvy tříd (API)	Všechny třídy zůstávají stejné v každé síti.	Všechny třídy zůstávají stejné v každé síti.

Tabulka 76. Rozdíly mezi knihovnami IBM MQ classes for .NET Framework a IBM MQ classes for .NET (.NET Standard a .NET 6) (pokračování)

Funkce	**IBM MQ classes for .NET Framework**	**IBM MQ classes for .NET (knihovny.NET Standard a .NET 6)**
Operační systém	Windows	Windows Kontejnery v nástroji Docker Linux macOS
soubor `app.config` (konfigurační soubor pro povolení trasování v redistribuovatelném klientovi)	Soubor `app.config` se používá k povolení trasování pro redistribuovatelný balík a samostatného klienta IBM MQ .NET . Další informace o proměnných, které používáte pro trasování, včetně **MQTRACEPATH** a **MQTRACELEVEL**, naleznete v tématu Trasování klienta IBM MQ classes for .NET Framework pomocí konfiguračního souboru aplikace .	`app.config` není podporován. Použijte proměnné prostředí.

Tabulka 76. Rozdíly mezi knihovnami IBM MQ classes for .NET Framework a IBM MQ classes for .NET (.NET Standard a .NET 6) (pokračování)

Funkce	IBM MQ classes for .NET Framework	IBM MQ classes for .NET (knihovny.NET Standard a .NET 6)
Trasovat	V případě úplné instalace klienta produktu IBM MQmůžete pomocí příkazu **strmqtrc** povolit trasování pro produkt IBM MQ classes for .NET Framework. Pro redistribuovatelné klienty se soubor `app.config` také používá k povolení trasování. Další informace viz Trasování IBM MQ .NET aplikace. V 9.3.3 V produktu IBM MQ 9.3.3můžete povolit nebo zakázat trasování pomocí souboru `mqclient.ini` a nastavením příslušných vlastností sekce Trasování. Můžete také dynamicky povolit a zakázat trasování pomocí souboru `mqclient.ini` . Další informace viz Trasování IBM MQ .NET aplikací s mqclient.ini.	Proměnná prostředí **MQDOTNET_TRACE_ON** se používá k povolení trasování pro redistribuovatelné klienty. Hodnoty menší nebo rovné 0 nepovolují trasování. Hodnota 1 povoluje trasování výchozí úrovně. Hodnota větší než 1 umožňuje podrobné trasování. Nastavení této proměnné prostředí na jinou hodnotu, jako je řetězec, nepovoluje trasování. Viz Trasování IBM MQ .NET aplikací používajících proměnné prostředí. Proměnná prostředí **MQDOTNET_TRACE_ON** kontroluje, zda je adresář trasování IBM MQ dostupný či nikoli. Je-li adresář trasování k dispozici, bude trasovací soubor vygenerován v adresáři trasování. Pokud však není nainstalován produkt IBM MQ , soubor trasování se zkopíruje do aktuálního pracovního adresáře. Další proměnné prostředí, včetně proměnných prostředí **MQERRORPATH**, **MQLOGLEVEL**, **MQSERVER**atd., které se používají pro IBM MQ classes for .NET Framework, lze použít a pracovat stejným způsobem. V 9.3.3 V produktu IBM MQ 9.3.3můžete povolit nebo zakázat trasování pomocí souboru `mqclient.ini` a nastavením příslušných vlastností sekce Trasování. Můžete také dynamicky povolit a zakázat trasování pomocí souboru `mqclient.ini` . Další informace viz Trasování IBM MQ .NET aplikací s mqclient.ini.
Přenosové režimy	Spravované, nespravované a vazby	Spravováno

Tabulka 76. Rozdíly mezi knihovnami IBM MQ classes for .NET Framework a IBM MQ classes for .NET (.NET Standard a .NET 6) (pokračování)

Funkce	IBM MQ classes for .NET Framework	IBM MQ classes for .NET (knihovny.NET Standard a .NET 6)
TLS	Úložiště klíčů Windows se používá k ukládání certifikátů.	Windows Úložiště klíčů Windows musí být použito k uložení certifikátů. Povolené hodnoty jsou *USER nebo *SYSTEM. Na základě vstupu se klient produktu IBM MQ .NET podívá do úložiště klíčů Windows aktuálního uživatele, nebo se podívá do celého systému. Linux Na systému Linux se k instalaci certifikátů doporučuje použít třídu X509Store a produkt .NET Core nainstaluje certifikáty do následujícího umístění: "`.dotnet/corefx/cryptography/x509stores`".
CCDT	Podporováno	Podporováno a nastavení cesty CCDT jsou stejná jako pro třídy .NET Framework.
Automatické znovupřipojení klienta	Podporováno	Podporováno
Distribuované transakce	Podporováno	Nepodporováno
Instalace dynamicky propojených knihoven (dll) do globální mezipaměti sestavení (GAC)	Knihovny Dll jsou nainstalovány do GAC jako součást instance IBM MQ.	Knihovny Dll nejsou nainstalovány do GAC jako součást instance IBM MQ.

Poznámka: Windows Identifikátory zabezpečení Windows (SID):

Ověření na úrovni domény není podporováno pro knihovny IBM MQ classes for .NET (.NET Standard a .NET 6). ID přihlášeného uživatele se používá pro ověření.

Vývoj aplikací IBM MQ .NET Core na systému macOS

macOS

Aplikace IBM MQ .NET Core lze vyvíjet na macOS.

Knihovny produktu IBM MQ .NET nejsou zabaleny se sadou nástrojů macOS , takže je musíte zkopírovat z klienta Windows nebo Linux IBM MQ do adresáře macOS. Tyto knihovny pak můžete použít k vývoji aplikací IBM MQ .NET Core na systému macOS.

Jakmile jsou tyto aplikace vyvinuty, lze je spouštět v prostředí Windows nebo Linux .

Související pojmy

“Instalace produktu IBM MQ classes for .NET Framework” na stránce 547
Produkt IBM MQ classes for .NET Framework, včetně ukázek, se instaluje s produktem IBM MQ. Je zde předpoklad Microsoft.NET Framework na Windows.

“Instalace produktu IBM MQ classes for XMS .NET” na stránce 602

Produkt IBM MQ classes for XMS .NET, včetně ukázek, se instaluje s produktem IBM MQ v systémech Windows a Linux.

Windows Linux Stažení IBM MQ classes for .NET z úložiště NuGet

IBM MQ classes for .NET je k dispozici ke stažení z úložiště NuGet , aby je mohli snadno využívat vývojáři produktu .NET .

Informace o této úloze

NuGet je správce balíků pro Microsoft vývojové platformy včetně .NET. Nástroje klienta NuGet poskytují schopnost vytvářet a spotřebovávat balíky. Balík NuGet je jediný komprimovaný soubor s příponou `.nupkg` , který obsahuje kompilovaný kód (DLL), další soubory související s tímto kódem a popisný soubor typu manifest, který obsahuje informace, jako je číslo verze balíku.

Balík `IBMMQDotnetClient` NuGet , který obsahuje knihovnu `amqmdnetstd.dll` , můžete stáhnout z galerie NuGet , která je centrálním úložištěm balíků používaným všemi autory a spotřebiteli balíků.

Poznámka: V 9.3.1 V produktu IBM MQ 9.3.1obsahuje balík NuGet knihovny sestavené pomocí .NET Standard 2.0 a .NET 6 jako cílový rámec. Knihovny pro produkt .NET Standard 2.0 jsou k dispozici ve složce `netstandard2.0` a knihovny pro produkt .NET 6 jsou k dispozici ve složce `net6.0` .

Existují tři způsoby stažení balíku `IBMMQDotnetClient` :

- Pomocí Microsoft Visual Studio. Produkt NuGet je distribuován jako rozšíření Microsoft Visual Studio . V produktu Microsoft Visual Studio 2012je produkt NuGet standardně předinstalován.
- Z příkazového řádku buď pomocí produktu NuGet Package Manager, nebo pomocí rozhraní příkazového řádku .NET .
- Pomocí webového prohlížeče.

Pokud jde o redistribuovatelný balík, povolíte trasování pomocí proměnné prostředí **`MQDOTNET_TRACE_ON`**.

Procedura

- Chcete-li stáhnout balík `IBMMQDotnetClient` pomocí uživatelského rozhraní správce balíků v produktu Microsoft Visual Studio, postupujte takto:
  a) Klepněte pravým tlačítkem myši na projekt .NET a poté klepněte na volbu **Spravovat balíky Nuget**.
  b) Klepněte na kartu **Procházet** a vyhledejte `"IBMMQDotnetClient"`.
  c) Vyberte balík a klepněte na tlačítko **Instalovat**.
  Během instalace poskytuje správce balíků informace o průběhu ve formě příkazů konzoly.
- Chcete-li stáhnout balík `IBMMQDotnetClient` z příkazového řádku, vyberte jednu z následujících voleb:
  - Pomocí správce balíků NuGet zadejte následující příkaz:

    ```
    Install-Package IBMMQDotnetClient -Version 9.1.4.0
    ```

    Během instalace poskytuje správce balíků informace o průběhu ve formě příkazů konzoly. Výstup můžete přesměrovat do souboru protokolu.
  - Pomocí rozhraní příkazového řádku .NET zadejte následující příkaz:

    ```
    dotnet add package IBMMQDotnetClient --version 9.1.4
    ```
- Pomocí webového prohlížeče stáhněte balík `IBMMQDotnetClient` z adresáře https://www.nuget.org/packages/IBMMQDotnetClient.

Související pojmy
IBM MQ Klient pro .NET informace o licenci

Související úlohy

“Stažení souboru IBM MQ classes for XMS .NET z úložiště NuGet” na stránce 605
Produkt IBM MQ classes for XMS .NET je k dispozici pro stažení z úložiště NuGet , aby jej mohli snadno využívat vývojáři produktu .NET .

Windows Instalace produktu IBM MQ classes for .NET Framework

Produkt IBM MQ classes for .NET Framework, včetně ukázek, se instaluje s produktem IBM MQ. Je zde předpoklad Microsoft.NET Framework na Windows.

Nejnovější verze produktu IBM MQ classes for .NET Framework se standardně instaluje jako součást standardní instalace produktu IBM MQ ve funkci *Java a .NET Messaging a Web Services* . Pokyny k instalaci naleznete v tématu Instalace serveru IBM MQ na systému Windows nebo Instalace klienta IBM MQ na systémech Windows.

V 9.3.0 V 9.3.0 Chcete-li spustit produkt IBM MQ classes for .NET Framework z adresáře IBM MQ 9.3.0, musíte nainstalovat produkt Microsoft.NET Framework V4.7.2 nebo novější. Jedná se o změnu z IBM MQ 9.2 , kde minimální požadovaná verze byla V4.6.2.

V 9.3.0 V 9.3.0 Existující aplikace, které jsou kompilovány pomocí produktu Microsoft.NET Framework V3.5 , lze spustit bez opětovné kompilace přidáním následující značky do souboru `app.config` aplikace:

```
<configuration>
 <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2"/>
 </startup>
</configuration>
```

Poznámka: Pokud není produkt Microsoft .NET Framework V4.7.2 nebo vyšší nainstalován před instalací produktu IBM MQ, instalace produktu IBM MQ pokračuje bez chyb, ale produkt IBM MQ classes for .NET není k dispozici. Pokud je.NET Framework nainstalován po instalaci produktu IBM MQ, musí být sestavení produktu IBM MQ.NET registrována spuštěním skriptu *WMQInstallDir*`\bin\amqiRegisterdotNet.cmd` , kde *WMQInstallDir* je adresář, kde je nainstalován produkt IBM MQ . Tento skript nainstaluje požadovaná sestavení do mezipaměti GAC (Global Assembly Cache). Sada souborů `amqi*.log` , které zaznamenávají provedené akce, jsou vytvořeny v adresáři %TEMP% . Skript `amqiRegisterdotNet.cmd` není nutné znovu spustit, pokud je produkt .NET upgradován na verzi V4.7.2 nebo vyšší z dřívější verze, například z verze .NET V3.5.

Pokud jste v prostředí s více instalačními prostředí dříve nainstalovali produkt IBM MQ classes for .NET jako podpůrný balík, nemůžete nainstalovat produkt IBM MQ , pokud nejprve neodinstalujete podpůrný balík. Funkce IBM MQ classes for .NET , která je nainstalována s produktem IBM MQ , obsahuje stejné funkce jako balík podpory.

K dispozici jsou také ukázkové aplikace, včetně zdrojových souborů; viz “Ukázkové aplikace pro .NET” na stránce 548.

Informace o použití vlastního kanálu IBM MQ pro Microsoft WCF s .NETviz “Vývoj aplikací Microsoft Windows Communication Foundation pomocí IBM MQ” na stránce 1222

Související pojmy

“Instalace produktu IBM MQ classes for .NET” na stránce 541
Produkt IBM MQ classes for .NET, včetně ukázek, se instaluje s operačním systému IBM MQ na systémech Windows a Linux .

Související úlohy

Trasování IBM MQ aplikací .NET

Volby pro připojení produktu IBM MQ classes for .NET ke správci front

Existují tři režimy připojení produktu IBM MQ classes for .NET ke správci front. Zvažte, který typ připojení nejlépe vyhovuje vašim požadavkům.

Připojení vazeb klienta

Chcete-li použít IBM MQ classes for .NET jako IBM MQ MQI client, můžete jej nainstalovat pomocí konzoly IBM MQ MQI clientbuď na počítač serveru IBM MQ , nebo na samostatný počítač. Připojení vazeb klienta může používat transakce XA nebo jiné než XA.

Připojení vazeb serveru

Při použití v režimu vazeb serveru produkt IBM MQ classes for .NET namísto komunikace prostřednictvím sítě používá rozhraní API správce front. To poskytuje lepší výkon pro aplikace IBM MQ než použití síťových připojení.

Chcete-li použít připojení vazeb, musíte nainstalovat produkt IBM MQ classes for .NET na server IBM MQ .

Připojení spravovaného klienta

Připojení provedené v tomto režimu se připojí jako klient IBM MQ k serveru IBM MQ spuštěnému na lokálním nebo vzdáleném počítači.

Připojení IBM MQ classes for .NET v tomto režimu zůstane ve spravovaném kódu .NET a nebude provádět žádná volání nativních služeb. Další informace o spravovaném kódu naleznete v dokumentaci k produktu Microsoft .

Použití spravovaného klienta má řadu omezení. Další informace o těchto informacích viz “Připojení spravovaných klientů” na stránce 563.

Ukázkové aplikace pro .NET

Chcete-li spustit vlastní aplikace .NET , postupujte podle pokynů pro verifikační programy a namísto ukázkových aplikací nahraďte název aplikace.

K dispozici jsou následující ukázkové aplikace:

- Aplikace pro vložení zprávy
- Aplikace pro získání zprávy
- Aplikace "Ahoj světe"
- Aplikace publikování/odběru
- Aplikace používající vlastnosti zprávy

Všechny tyto ukázkové aplikace jsou dodávány v jazyce C# a některé také v jazycích C++ a Visual Basic. Aplikace můžete psát v libovolném jazyce podporovaném produktem .NET.

Program SPUT "Vložit zprávu" (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 (volitelné)
- Definice kanálu (volitelné), například SYSTEM.DEF.SVRCONN/TCP/hostname(1414)

Není-li zadán žádný název správce front, použije se výchozí lokální správce front. Je-li kanál definován, má stejný formát jako proměnná prostředí MQSERVER.

"Získat zprávu" 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 (volitelné)
- Definice kanálu (volitelné), například SYSTEM.DEF.SVRCONN/TCP/hostname(1414)

Není-li zadán žádný název správce front, použije se výchozí lokální 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 (volitelné)
- Definice kanálu (volitelné), 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, použije se výchozí lokální správce front.

Program "publikování/odběr" (MQPubSubSample.cs)
Tento program ukazuje, jak používat produkt IBM MQ publikování/odběr. Dodává se pouze v jazyce C#. Program má dva parametry:

- Název správce front (volitelné)
- Definice kanálu (volitelné)

Program "Vlastnosti zprávy" (MQMessagePropertiesSample.cs)
Tento program ukazuje, jak používat vlastnosti zprávy. Dodává se pouze v jazyce C#. Program má dva parametry:

- Název správce front (volitelné)
- Definice kanálu (volitelné)

Instalaci můžete ověřit kompilací a spuštěním těchto aplikací.

Umístění instalace

Ukázkové aplikace se instalují do následujících umístění podle jazyka, ve kterém 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

Spravovaná 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

Sestavení ukázkových aplikací

Chcete-li sestavit ukázkové aplikace, je pro každý jazyk dodán dávkový soubor.

C#

MQ_INSTALLATION_PATH\Tools\dotnet\samples\cs\bldcssamp.bat

Soubor bldcssamp.bat obsahuje řádek pro každou ukázku, což je vše, co je nezbytné pro sestavení tohoto ukázkového programu:

```
csc /t:exe /r:System.dll /r:amqmdnet.dll /lib: MQ_INSTALLATION_PATH\bin
/out:nmqwrld.exe nmqwrld.cs
```

Spravovaná C++

MQ_INSTALLATION_PATH\Tools\dotnet\samples\mcp\bldmcpsamp.bat

Soubor bldmcpsamp.bat obsahuje řádek pro každou ukázku, což je vše, co je nezbytné pro sestavení tohoto ukázkového programu:

```
cl /clr:oldsyntax MQ_INSTALLATION_PATH\bin mmqwrld.cpp
```

Chcete-li kompilovat tyto aplikace 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 každou ukázku, což je vše, co je nezbytné pro sestavení tohoto ukázkového programu:

```
vbc /r:System.dll /r: MQ_INSTALLATION_PATH\bin\amqmdnet.dll /out:vmqwrld.exe vmqwrld.vb
```

Ukázky pro použití produktu IBM MQ s produktem Microsoft .NET Core

V produktu IBM MQ 9.2.0 IBM MQ podporuje .NET Core pro aplikace IBM MQ .NET v prostředí Windows . Produkt IBM MQ classes for .NET Standard, včetně ukázek, se standardně instaluje jako součást standardní instalace produktu IBM MQ .

Ukázkové aplikace pro produkt IBM MQ .NET jsou nainstalovány v adresáři `&MQINSTALL_PATH&/samp/dotnet/samples/cs/core/base`. K dispozici je také skript, který lze použít ke kompilaci ukázek.

Ukázky můžete sestavit pomocí dodaných souborů `build.bat` . Pro každou ukázku v následujícím umístění v systému Windowsexistuje jeden soubor `build.bat` :

- `MQ\tools\dotnet\samples\cs\core\base\SimpleGet`
- `MQ\tools\dotnet\samples\cs\core\base\SimplePut`

Linux V produktu IBM MQ 9.2.0produkt IBM MQ také podporuje jádro pro aplikace v prostředích Linux .

Další informace o použití produktu IBM MQ s produktem Microsoft .NET Coreviz “Instalace produktu IBM MQ classes for .NET” na stránce 541.

Konfigurace správce front tak, aby přijímal připojení klienta TCP/IP

Konfigurujte 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 připojení klienta TCP/IP. V případě produkčního systému musíte při konfiguraci správců front také zvážit dopady na 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: Tato ukázka je určena pouze pro použití v prostředí sandboxu, protože nezahrnuje žádné posouzení dopadů na zabezpečení. V případě produkčního systému zvažte použití TLS nebo uživatelské procedury zabezpečení. Další informace viz Zabezpečení IBM MQ .
2. Spustit modul listener:

```
runmqlsr -t tcp [-m qmnqme ] [-p portnum ]
```

   Poznámka: Hranaté závorky označují volitelné parametry; hodnota *qmname* není pro výchozího správce front vyžadována a číslo portu *portnum* není vyžadováno, pokud používáte výchozí hodnotu (1414).

Distribuované transakce v adresáři .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 nebo více správci prostředků.

Transakce mohou být buď jednofázový, nebo dvoufázový proces potvrzování. Jednofázové potvrzování je proces, v němž se transakce účastní pouze jeden správce prostředků a v procesu dvoufázového potvrzování existuje více správců prostředků, kteří se transakce účastní. V procesu dvoufázového potvrzování odešle správce transakcí volání prepare, aby zkontroloval, zda jsou všichni správci prostředků připraveni na potvrzení. Když obdrží potvrzení od všech správců prostředků, je vydáno volání potvrzení. Jinak dojde k odvolání celé transakce. Další podrobnosti viz Správa transakcí a podpora . Správci prostředků by měli informovat správce transakcí o jejich účasti na transakci. Když správce prostředků informuje správce transakcí o své účasti, obdrží od správce transakcí zpětná volání při potvrzení nebo odvolání transakce.

Třídy produktu IBM MQ .NET již podporují distribuované transakce v nespravovaných připojeních a připojeních v režimu vazeb serveru. V těchto režimech třídy IBM MQ .NET delegují všechna svá volání na klienta rozšířené transakce C, který spravuje zpracování transakcí jménem .NET.

Třídy produktu IBM MQ.NET nyní podporují distribuované transakce ve spravovaném režimu, kde IBM MQ .NET Třídy používají pro podporu distribuovaných transakcí obor názvů System.Transactions . Infrastruktura System.Transactions usnadňuje a zefektivňuje programování transakcí prostřednictvím podpory transakcí zahájených ve všech správcích prostředků včetně IBM MQ. Aplikace IBM MQ .NET může vkládat a získávat zprávy pomocí .NET implicitního programování transakcí nebo explicitního programovacího modelu transakcí. V implicitních transakcích jsou hranice transakcí vytvořeny aplikačním programem, který rozhoduje o tom, kdy transakci potvrdit, odvolat (v případě explicitních transakcí) nebo dokončit. V explicitních transakcích musíte explicitně určit, zda chcete transakci potvrdit, odvolat a dokončit.

IBM MQ.NET používá Microsoft koordinátora distribuovaných transakcí (MS DTC) jako správce transakcí, který koordinuje a spravuje transakce mezi více správci prostředků. IBM MQ se používá jako správce

prostředků. Mějte na paměti, že s transakcemi XA nelze používat protokol TLS. Musíte použít CCDT. Další informace naleznete v tématu Použití rozšířeného transakčního klienta s kanály TLS.

Produkt IBM MQ.NET se řídí modelem zpracování distribuovaných transakcí X/Open (DTP). Model zpracování distribuovaných transakcí X/Open je model zpracování distribuovaných transakcí navržený konsorciem dodavatelů Open Group. Tento model je standardem mezi většinou komerčních dodavatelů v oblasti zpracování transakcí a databázových domén. Většina produktů pro správu obchodních transakcí podporuje model X/DTP.

Způsoby transakcí

- "Distribuované transakce ve spravovaném režimu .NET" na stránce 553
- Distribuované transakce pro nespravovaný režim

Koordinace transakcí v různých scénářích

- Připojení se může účastnit několika transakcí, ale v každém okamžiku je aktivní pouze jedna transakce.
- Během transakce MQQueueManager.Volání odpojení je uznáno. V tomto případě je transakce požádána o odvolání.
- Během transakce je respektováno volání MQQueue.Close nebo MQTopic.Close . V tomto případě je transakce požádána o odvolání.
- Hranice transakcí vytváří aplikační program, který rozhoduje, kdy transakci potvrdit, odvolat (pro explicitní transakce) nebo dokončit (pro implicitní transakce).
- Pokud se aplikace klienta přeruší během transakce s neočekávanou chybou před vyvoláním volání Put nebo Get pro volání fronty nebo tématu, transakce se odvolá a dojde k výjimce MQException.
- Pokud je během volání Put nebo Get ve frontě nebo volání tématu vrácen kód příčiny MQCC_FAILED, dojde k výjimce MQException s kódem příčiny a transakce je odvolána. Pokud již správce transakcí vydal volání přípravy, produkt IBM MQ .NET vrátí požadavek na přípravu vynucením odvolání transakce. Pak správce transakcí DTC způsobí odvolání aktuální práce se všemi správci prostředků v aktuálních okolních transakcích.
- V průběhu transakce zahrnující více správců prostředků v případě, že z nějakého důvodu prostředí způsobí pozastavení volání Put nebo Get na dobu neurčitou, správce transakcí čeká na stanovenou dobu. Po uplynutí časového limitu dojde k odvolání veškeré aktuální práce se všemi správci prostředků v aktuálních okolních transakcích. Dojde-li během přípravné fáze k tomuto neomezenému čekání, může správce transakcí vypršel časový limit nebo vyvolat neověřené volání prostředku. V takovém případě je transakce odvolána.
- Aplikace používající transakce musí vkládány nebo získat zprávy pod SYNC_POINT. Pokud je volání Put nebo Get zprávy vydáno pod transakčním kontextem, který není pod SYNC_POINT, volání selže s kódem příčiny MQRC_UNIT_OF_WORK_NOT_STARTED.

Rozdíly v chování mezi podporou transakcí spravovaného a nespravovaného klienta pomocí oboru názvů Microsoft.NET System.Transactions

Vnořené transakce mají TransactionScope uvnitř jiného TransactionScope

- IBM MQ .NET plně spravovaný klient podporuje vnořený TransactionScope
- IBM MQ .NET nespravovaný klient nepodporuje vnořený TransactionScope

Závislé transakce z System.Transactions

- IBM MQ .NET plně spravovaný klient podporuje mechanizmus závislých transakcí poskytovaný produktem System.Transactions.
- IBM MQ .NET nespravovaný klient nepodporuje mechanizmus závislých transakcí poskytovaný produktem System.Transactions.

Ukázky produktů

Ukázky produktu `SimpleXAPuta SimpleXAGet` jsou k dispozici v části `WebSphere MQ\tools\dotnet\samples\cs\base`. Ukázky jsou aplikace v jazyce C#, které demonstrují použití příkazů MQPUT a MQGET v rámci distribuovaných transakcí s použitím oboru názvů SystemTransactions . Další informace o těchto ukázkách viz "Vytvoření jednoduchého vložení a získání zpráv v rámci TransactionScope" na stránce 556.

Distribuované transakce ve spravovaném režimu .NET

Třídy IBM MQ .NET používají obor názvů System.Transactions pro podporu distribuovaných transakcí ve spravovaném režimu. Ve spravovaném režimu koordinuje služba MS DTC a spravuje distribuované transakce na všech serverech uvedených v transakci.

Třídy produktu IBM MQ .NET poskytují explicitní programovací model založený na třídě System.Transactions.Transaction a implicitní programovací model pomocí třídy System.Transactions.TransactionScope, kde jsou transakce automaticky spravovány infrastrukturou.

Implicitní transakce

Následující část kódu popisuje, jak aplikace IBM MQ .NET vkládá zprávu pomocí .NET implicitního programování transakcí.

```
Using (TransactionScope scope = new TransactionScope ())
{
    Q.Put (putMsg,pmo);
    scope.Complete ();
}

Q.close();
qMgr.Disconect();}
```

Vysvětlení toku kódu implicitní transakce

Kód vytvoří *TransactionScope* a vloží zprávu do rozsahu. Poté zavolá příkaz *Dokončit* a informuje koordinátora transakce o dokončení transakce. Koordinátor transakcí nyní vydá příkaz *prepare* a *commit* pro dokončení transakce. Pokud je zjištěn problém, zavolá se *rollback* .

Explicitní transakce

Následující kód popisuje, jak aplikace IBM MQ .NET vkládá zprávy pomocí .NET explicitního programovacího modelu transakcí.

```
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 vytváří transakci pomocí třídy *CommitableTransaction* . Vloží zprávu do tohoto rozsahu a pak explicitně volá příkaz *commit* k dokončení transakce. Pokud se vyskytli nějaké problémy, zavolá se *rollback* .

Distribuované transakce v nespravovaném režimu .NET

Třídy produktu IBM MQ.NET podporují nespravovaná připojení (klient) s použitím rozšířeného klienta transakcí a modelu COM + /MTS jako koordinátora transakcí s použitím implicitního nebo explicitního modelu programování transakcí. V nespravovaném režimu třídy IBM MQ .NET delegují všechna svá volání na klienta rozšířené transakce C, který spravuje zpracování transakcí jménem .NET.

Zpracování transakcí je řízeno externím správcem transakcí, který koordinuje globální jednotku práce pod kontrolou rozhraní API správce transakcí. Příkazy MQBEGIN, MQCMIT a MQBACK nejsou k dispozici. IBM MQ Třídy .NET vystavují tuto podporu prostřednictvím svého nespravovaného režimu přenosu (klient C). Viz Konfigurace správců transakcí vyhovujících standardu XA

MTS se vyvíjí jako systém zpracování transakcí (TP), aby poskytoval stejné funkce v systému Windows NT , jaké jsou k dispozici v systémech CICS, Tuxedo a na jiných platformách. Po instalaci MTS je do produktu Windows NT přidána samostatná služba s názvem Microsoft Distributed Transaction Coordinator (MSDTC). MSDTC koordinuje transakce, které zahrnují oddělená datová úložiště nebo prostředky. K tomu, aby fungoval, vyžaduje, aby každé datové úložiště implementovalo svého vlastního proprietárního správce prostředků.

Produkt IBM MQ se stane kompatibilní s produktem MSDTC implementací rozhraní (proprietární rozhraní správce prostředků), ve kterém spravuje mapování volání DTC XA na volání IBM MQ(X/Open). IBM MQ hraje roli správce prostředků.

Když komponenta, jako např. COM +, požádá o přístup k serveru IBM MQ, COM obvykle zkontroluje s příslušným kontextovým objektem MTS, zda je vyžadována transakce. Je-li vyžadována transakce, COM informuje koordinátor DTC a automaticky spustí pro tuto operaci integrální transakci IBM MQ . Pak COM pracuje s daty prostřednictvím softwaru MQMTS, vkládání a získávání zpráv podle potřeby. Instance objektu získaná z modelu COM volá metodu SetComplete nebo SetAbort po ukončení všech akcí s daty. Když aplikace vydá příkaz SetComplete, volání signalizuje koordinátoru DTC, že aplikace dokončila transakci a koordinátor DTC může pokračovat v procesu dvoufázového potvrzování. Koordinátor DTC poté vydá volání MQMTS, které následně vyvolá volání IBM MQ za účelem potvrzení nebo odvolání transakce.

Zápis aplikace IBM MQ .NET pomocí nespravovaného klienta

Chcete-li spustit v kontextu modelu COM +, musí třída .NET dědit ze systému.EnterpriseServices.ServicedComponent. Pravidla a doporučení pro vytvoření sestavení, která používají servisované komponenty, jsou následující:

Poznámka: Následující kroky jsou relevantní pouze v případě, že používáte režim System.EnterpriseServices .

- Třída a metoda spouštěná v modelu COM + musí být veřejné (žádné interní třídy a žádné chráněné nebo statické metody).
- Atributy třídy a metody: Atribut TransactionOption určuje úroveň transakce třídy, tj. zda jsou transakce zakázány, podporovány nebo požadovány. Atribut AutoComplete v metodě ExecuteUOW() instruuje COM + k potvrzení transakce, pokud není vyvolána žádná neošetřená výjimka.
- Silné pojmenování sestavy: Sestava musí být silně pojmenovaná a registrovaná v globální mezipaměti sestavení (GAC). Sestava je registrována v COM + explicitně nebo línou registrací poté, co je registrována v GAC.
- Registrace sestavení v modelu COM +: Připravte sestavení tak, aby bylo vystaveno klientům modelu COM. Poté vytvořte knihovnu typů pomocí nástroje pro registraci sestavení regasm.exe.

  ```
  regasm UnmanagedToManagedXa.dll
  ```

- Zaregistrujte sestavu do GAC `gacutil /i UnmanagedToManagedXa.dll`.
- Zaregistrujte sestavení v COM + pomocí instalačního nástroje služeb .NET , regsvcs.exe. Viz knihovna typů vytvořená příkazem regasm.exe:

  ```
  Regsvcs /appname:UnmanagedToManagedXa /tlb:UnmanagedToManagedXa.tlb  UnmanagedToManagedXa.dll
  ```

- Sestava je nasazena do GAC a později je registrována v COM + pomocí pomalé registrace. Rámec .NET se postará 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();
}
```

Vytvoření jednoduchého vložení a získání zpráv v rámci TransactionScope

Ukázkové aplikace C# produktu jsou k dispozici v produktu IBM MQ. Tyto jednoduché aplikace demonstrují vkládání a získávání 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

Příkladem je jednoduchá aplikace `SimpleXAPut` a `SimpleXAGet`. Programy `SimpleXAPut` a `SimpleXAGet` jsou aplikace v jazyce C# dostupné v rámci produktu IBM MQ. Produkt `SimpleXAPut` demonstruje použití příkazu MQPUT v části Distribuované transakce s použitím oboru názvů SystemTransactions . `SimpleXAGet` demonstruje použití MQGET v části Distribuované transakce používající obor názvů SystemTransactions .

`SimpleXAPut` se nachází v adresáři `MQ\tools\dotnet\samples\cs\base`

Postup

Aplikace lze spustit s parametry příkazového řádku z `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 se jednat o frontu nebo téma. Pro frontu zadejte hodnotu `queue://queueName` a pro téma zadejte hodnotu `topic://topicName`.

`-host`

Může se jednat o název hostitele, například localhost nebo adresu IP.

`-port`

Port, na kterém je spuštěn správce front.

`-channel`

Používaný kanál připojení. Výchozí nastavení je `SYSTEM.DEF.SVRCONN`

`-transaction`

Výsledek transakce, například potvrzení nebo odvolání.

`-mode`

Režim přenosu, 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
```

Obnovení transakcí v produktu IBM MQ .NET

Tento oddíl popisuje proces obnovy transakcí v produktu IBM MQ .NET XA pomocí spravovaného režimu.

Informace o této úloze

Při zpracování distribuovaných transakcí lze transakce úspěšně dokončit, ale mohou existovat scénáře, ve kterých může dojít k selhání transakce z mnoha příčin. Tyto příčiny mohou zahrnovat selhání systému, selhání hardwaru, síťovou chybu, nesprávná nebo neplatná data, chyby aplikace nebo přírodní nebo člověkem způsobené katastrofy. Není možné zabránit selhání transakcí. Distribuovaný transakční systém musí být schopen zpracovat tato selhání. Musí být schopen detekovat a opravit chyby, když se vyskytnou. Tento proces se nazývá Obnova transakce.

Důležitým aspektem zpracování distribuovaných transakcí je obnova neúplných nebo nejistých transakcí. Je nezbytné spustit obnovu, protože část transakce Unit of Work je uzamčena, dokud není obnovena. Microsoft.NET z knihovny tříd System.Transactions poskytuje volbu pro obnovu neúplných/neověřených transakcí. Tato podpora obnovy očekává, že Resource Manager bude udržovat transakční protokoly a spouštět zotavení v případě potřeby.

V modelu zotavení transakcí produktu Microsoft .NET správce transakcí (System.Transactionsnebo Microsoft Distributed Transaction Coordinator (MS DTC) nebo obojí) iniciuje, koordinuje a řídí obnovu transakcí. Správci prostředků založené na protokolu OLE Tx (protokol Microsoft XA) poskytují volby pro konfiguraci koordinátora DTC pro řízení, koordinaci a řízení jejich obnovy. Za tímto účelem musí správci prostředků registrovat přepínač XA_Switch v produktu MS DTC pomocí nativního rozhraní.

Přepínač XA_Switch poskytuje koordinátorovi distribuovaných transakcí vstupní body funkcí XA, jako např. xa_start, xa_end a xa_recover ve Resource Manager .

Obnova pomocí koordinátora distribuovaných transakcí (DTC) produktu Microsoft :

Microsoft Koordinátor distribuovaných transakcí poskytuje dva druhy procesů obnovy.

Studená obnova

Studené zotavení se provádí v případě, že dojde k selhání procesu správce transakcí v době, kdy je otevřeno připojení ke správci prostředků XA. Po restartování správce transakcí načte protokoly správce transakcí a znovu naváže připojení ke správci prostředků XA a poté zahájí zotavení.

Zotavení za běhu

Zotavení za běhu se provádí v případě, že správce transakcí zůstane v době, kdy dojde k selhání připojení mezi správcem transakcí a správcem prostředků XA z důvodu selhání správce prostředků XA nebo sítě. Po selhání se správce transakcí pravidelně pokouší znovu připojit ke správci prostředků XA. Po opětovném vytvoření připojení správce transakcí zahájí 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. Jediným rozdílem je obnova transakce. System.Transactions očekává, že správci prostředků budou řídit obnovu sami a poté budou koordinovat se správci transakcí (MS DTC). Resource Manager musí požádat o obnovu konkrétní nedokončené transakce a poté ji správce transakcí přijme a koordinuje na základě skutečného výsledku dané transakce.

Proces obnovy transakcí pro produkt IBM MQ .NET

Tento oddíl popisuje, jak lze obnovit distribuované transakce pomocí tříd IBM MQ .NET .

Přehled

Chcete-li obnovit nedokončenou transakci, jsou požadovány informace o zotavení. Správci prostředků musí protokolovat informace o zotavení transakce do úložiště. IBM MQ Třídy .NET následují podobnou cestu. Informace o zotavení transakce jsou protokolovány do systémové fronty s názvem SYSTEM.DOTNET.XARECOVERY.QUEUE.

Obnova transakcí v produktu IBM MQ .NET je dvoufázový proces:

1. Protokolování informací o zotavení transakcí v systému SYSTEM.DOTNET.XARECOVERY.QUEUE.
2. Obnova transakcí pomocí aplikace monitoru XA WmqDotnetXAMonitor.

SYSTEM.DOTNET.XARECOVERY.QUEUE

SYSTEM.DOTNET.XARECOVERY.QUEUE je systémová fronta, která uchovává informace o obnově transakcí pro nedokončené transakce. Tato fronta je vytvořena při vytvoření správce front.

Pro každou transakci je během přípravné fáze do systému SYSTEM.DOTNET.XARECOVERY.QUEUE. Zpráva se odstraní, pokud je volání potvrzení úspěšné.

Poznámka: Nesmíte odstranit SYSTEM.DOTNET.XARECOVERY.QUEUE .

Aplikace WMQDotnetXAMonitor

IBM MQ .NET Aplikace XA Monitor WmqDotnetXAMonitor je aplikace spravovaná produktem .NET , která monitoruje správce front a zpracovává zprávy v systému SYSTEM.DOTNET.XARECOVERY.QUEUE a obnoví nedokončené transakce

Pokud agent kanálu zpráv (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 odpověď určené v původní zprávě. (Pokud se fronta pro odpověď nachází ve stejném správci front jako agent MCA, zpráva se vloží přímo do této fronty, nikoli do přenosové fronty.)

Za nedokončené transakce se považují následující transakce, které jsou získány zpět:

- Pokud je transakce připravena, ale příkaz COMMIT nebyl dokončen během časového limitu.
- Pokud je transakce připravena, ale správce front IBM MQ byl vypnutý.
- Pokud je transakce připravena, ale pak se správce transakcí vypnul.

  Aplikace XA Monitor musí být spuštěna ze stejného systému, kde je spuštěna klientská aplikace IBM MQ .NET . Pokud existují aplikace, které jsou spuštěny na více systémech a připojují se ke stejnému správci front, musí být aplikace WmqDotnetXAMonitor spuštěna ze všech systémů. Ačkoli má každý klientský počítač spuštěnou instanci aplikace monitoru XA pro obnovu aplikace, každá instance monitoru XA by měla být schopna identifikovat zprávu odpovídající transakci, kterou koordinoval lokální koordinátor MS DTC aktuálního monitoru XA, aby ji mohl znovu zapsat a dokončit.

Související pojmy

"Případy použití obnovy transakcí pro produkt IBM MQ .NET" na stránce 558
Existuje několik různých případů použití, ze kterých může být nutné transakce obnovit.

Související úlohy

"Použití aplikace WMQDotnetXAMonitor" na stránce 560
Klient IBM MQ .NET poskytuje aplikaci monitoru XA, WmqDotnetXAMonitor, kterou můžete použít k obnovení nedokončených distribuovaných transakcí. Aplikace WmqDotnetXAMonitor vytvoří připojení ke správci front, kde jsou transakce v nejistém stavu, a poté transakci vyřeší na základě nastavených parametrů.

Případy použití obnovy transakcí pro produkt IBM MQ .NET

Existuje několik různých případů použití, ze kterých může být nutné transakce obnovit.

- **IBM MQ Aplikace používající jeden koordinátor DTC a jednu instanci správce front:** V tomto příkladu použití při připojení ke správci front a spuštění transakceUoW(Unit of Work) a v případě, že transakce selže a stane se neúplnou, aplikace monitoru XA obnoví transakci a dokončí ji.

  V tomto případě použití bude spuštěna jediná instance aplikace monitoru XA, protože k transakcím je přidružen jeden správce front.
- **Více IBM MQ aplikací používajících jednu instanci koordinátora DTC a jednoho správce front:** V tomto případě použití existuje více než jedna aplikace IBM MQ v rámci jednoho koordinátoru DTC a všechny se připojují ke stejnému správci front a spouští UoW v rámci transakcí.

  Pokud transakce selžou a stanou se neúplnými, aplikace monitoru XA je obnoví a dokončí transakce týkající se všech aplikací.

  V tomto případě použití je spuštěna jediná instance aplikace monitoru XA, protože v transakcích se používá jeden správce front.
- **Více IBM MQ aplikací, více DTC, různé instance správce front:** V tomto příkladu použití existuje více než jedna aplikace IBM MQ v různých DTC (tj. každá aplikace je spuštěna v jiném počítači) a připojuje se k různým správcům front.

  Dojde-li k selhání a transakce se stane neúplnou, aplikace monitorování zkontroluje TransactionManagerve zprávě a určí adresu koordinátora DTC. Pokud se hodnota TransactionManagerWhereabouts shoduje s adresou DTC, pod kterou je monitor spuštěn, dokončí obnovu, jinak bude pokračovat v hledání, dokud nebude nalezena zpráva odpovídající jejímu DTC.

  V tomto případě použití bude pro každého klienta (uživatele nebo počítač) spuštěna pouze jedna instance aplikace monitoru XA, protože každý klient má svého vlastního správce front používaného v transakcích.
- **Více IBM MQ aplikací, více DTC, více stejných instancí správce front:** V tomto případě použití existuje více než jedna aplikace IBM MQ pod různými DTC (tj. každá aplikace je spuštěna na jiném počítači) a všechny se připojují ke stejnému správci front.

  Dojde-li k selhání a transakce se stane neúplnou, aplikace monitoru ověří TransactionManager, co se ve zprávě objeví, aby zkontrolovala, zda se adresa DTC a hodnota shodují s DTC, pod kterým je monitor spuštěn. Pokud se obě hodnoty shodují, dokončí obnovu, jinak pokračuje v hledání, dokud nenajde zprávu odpovídající svému DTC.

  V tomto případě použití bude pro každého klienta (uživatele nebo počítač) spuštěna pouze jedna instance aplikace XA Monitor, protože každý klient má své vlastní přidružení správce front používané v transakcích.
- **Více IBM MQ aplikací, jeden koordinátor DTC, různé instance správce front:** V tomto případě použití existuje více než jedna aplikace IBM MQ v rámci jednoho koordinátoru DTC (tj. v počítači je spuštěno více aplikací IBM MQ) a připojují se k různým správcům front.

  Pokud transakce selže a stane se neúplnou, aplikace monitoru obnoví transakci.

  V tomto případě použití bude spuštěn stejný počet instancí aplikace monitoru jako správci front, ke kterým jsou připojeni, protože každá aplikace má svého vlastního správce front používaného v transakcích a každý z nich musí být obnoven.

  Poznámka: Pokud není aplikace monitoru XA spuštěna na pozadí, můžete ji spustit.

Související pojmy

“Proces obnovy transakcí pro produkt IBM MQ .NET” na stránce 557
Tento oddíl popisuje, jak lze obnovit distribuované transakce pomocí tříd IBM MQ .NET .

Související úlohy

“Použití aplikace WMQDotnetXAMonitor” na stránce 560
Klient IBM MQ .NET poskytuje aplikaci monitoru XA, WmqDotnetXAMonitor, kterou můžete použít k obnovení nedokončených distribuovaných transakcí. Aplikace WmqDotnetXAMonitor vytvoří připojení ke správci front, kde jsou transakce v nejistém stavu, a poté transakci vyřeší na základě nastavených parametrů.

Použití aplikace WMQDotnetXAMonitor

Klient IBM MQ .NET poskytuje aplikaci monitoru XA, WmqDotnetXAMonitor, kterou můžete použít k obnovení nedokončených distribuovaných transakcí. Aplikace WmqDotnetXAMonitor vytvoří připojení ke správci front, kde jsou transakce v nejistém stavu, a poté transakci vyřeší na základě nastavených parametrů.

Informace o této úloze

Aplikace WMQDotnetXAMonitor musí být spuštěna ručně. Lze jej spustit kdykoli. Můžete jej spustit, když uvidíte zprávy v systému SYSTEM.DOTNET.XARECOVERY.QUEUE nebo ji můžete ponechat spuštěnou na pozadí před provedením jakékoli transakční práce s aplikacemi, které jsou napsány pomocí tříd IBM MQ .NET .

Hodnoty parametrů pro WMQDotnetXAMonitor můžete nastavit buď pomocí příkazového řádku, nebo pomocí konfiguračního souboru aplikace. Hodnoty poskytnuté prostřednictvím konfiguračního souboru aplikace mají přednost před hodnotami nastavenými prostřednictvím příkazového řádku.

Před produktem IBM MQ 9.3.0je připojení, které produkt WMQDotnetXAMonitor navazuje, nezabezpečené připojení.

V 9.3.0 V produktu IBM MQ 9.3.0máte možnost vytvořit zabezpečené připojení ke správci front nastavením dalších parametrů pro WMQDotnetXAMonitor.

Procedura

- Chcete-li poskytnout vstup pro WmqDotNETXAMonitor pomocí konfiguračního souboru aplikace, viz “Nastavení konfiguračního souboru aplikace WmqDotNETXAMonitor” na stránce 562.
- Chcete-li spustit aplikaci WMQDotnetXAMonitor z příkazového řádku, použijte následující příkaz s požadovanými parametry:

  Před IBM MQ 9.3.0:

  ```
  WmqDotnetXAMonitor.exe -m QueueManagerName -n ConnectionName -c ChannelName -i
  ```

  V 9.3.0 Z adresáře IBM MQ 9.3.0:

  ```
  WmqDotnetXAMonitor.exe -m QueueManagerName -n ConnectionName -c ChannelName -i -k SSL Key
  Repository -s Cipher Spec
  ```

  Parametry, které můžete zadat, jsou následující:

  – **-m *QueueManager***
  Název správce front.

  Volitelné

  -n *ConnectionName*
  Název připojení ve formátu hostitele (port). *ConnectionName* může obsahovat více než jeden název připojení. Více názvů připojení musí být uvedeno v seznamu odděleném čárkami, například `localhost (1414), localhost (1415), localhost (1416)`. Aplikace WMQDotnetXAMonitor spustí obnovu pro každý z názvů připojení uvedených v seznamu odděleném čárkami.

  -c *ChannelName*
  Název kanálu.

  -i
  Heuristické dokončení větve.

  Volitelné

V 9.3.0 **-k *Úložiště klíčů SSL***

Název úložiště klíčů SSL. Podporované hodnoty jsou:

- *SYSTEM (toto je výchozí hodnota)
- *USER

Volitelné

V 9.3.0 **-s *Specifikace šifrování***

CipherSpec , kterou nastavíte, musí být jednou ze specifikací CipherSpecs pro podporovanou verzi a může být nejlépe stejná jako ta, která je určena v zásadě skupiny Windows . Další informace viz téma “Podpora CipherSpec pro spravovaného klienta .NET” na stránce 582.

Povinné pro vytvoření zabezpečeného připojení ke správci front.

V 9.3.0 **-dn *Název SSLPeer***

Název partnera SSL použitý ke kontrole rozlišujícího názvu (DN) certifikátu ze správce front typu peer.

Volitelné

V 9.3.0 **-cl *Označení certifikátu***

Název popisku, který identifikuje certifikát.

Volitelné

V 9.3.0 **-sn *OutboundSNI***

Zda má být SNI (Server Name Indication) nastaven na název cílového kanálu IBM MQ pro vzdálený systém při inicializaci připojení TLS, nebo na název hostitele. Podporované hodnoty pro tuto volbu jsou:

- CHANNEL (toto je výchozí hodnota)
- HOSTNAME
- *

Není-li nastavena žádná hodnota, použije se výchozí hodnota, tj. CHANNEL.

Volitelné

V 9.3.0 **-cr *Kontrola odvolání certifikátu***

Zda se má provést kontrola odvolání certifikace. Podporované hodnoty pro tuto volbu jsou:

- ano
- false (toto je výchozí hodnota)

Volitelné

V 9.3.0 **-kr *KeyResetPočet***

Celkový počet nešifrovaných bajtů, které jsou odesílány a přijímány kanálem, než je znovu vyjednán tajný klíč použitý pro šifrování.

Výchozí hodnota 0 označuje, že tajné klíče nejsou nikdy znovu vyjednány

Volitelné

Aplikace WMQDotnetXAMonitor provádí následující akce:

1. Zkontroluje hloubku fronty SYSTEM.DOTNET.XARECOVERY.QUEUE v intervalu 100 sekund.
2. Je-li hloubka fronty větší než nula, vyhledá zprávy ve frontě a zkontroluje, zda zprávy splňují kritéria neúplné transakce.
3. Pokud zpráva splňuje neúplná kritéria transakce, stáhne ji a načte informace o zotavení transakce.
4. Určuje, zda se informace o zotavení vztahují k lokálnímu Microsoft koordinátoru distribuovaných transakcí (MS DTC). V takovém případě bude WMQDotnetXAMonitor pokračovat v obnově transakce, jinak se vrátí k procházení další zprávy.

5. Provede volání správce front za účelem zotavení nedokončené transakce.

Nastavení konfiguračního souboru aplikace WmqDotNETXAMonitor
Vstup pro aplikaci WmqDotNETXAMonitor produktu IBM MQ .NET XA Monitor můžete zadat pomocí konfiguračního souboru aplikace. Konfigurační soubor ukázkové aplikace je dodáván s produktem IBM MQ .NET. Tento ukázkový soubor můžete upravit podle svých požadavků.

Vstupní hodnoty poskytnuté prostřednictvím konfiguračního souboru aplikace mají nejvyšší prioritu. Zadáte-li vstupní hodnoty jak na příkazovém řádku, jak je popsáno v části “Použití aplikace WMQDotnetXAMonitor” na stránce 560 , tak v konfiguračním souboru aplikace, budou mít přednost hodnoty z konfiguračního souboru aplikace.

Ukázkový konfigurační soubor aplikace pro před IBM MQ 9.3.0.

```
<?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>
```

V 9.3.0 Ukázkový konfigurační soubor aplikace z adresáře IBM MQ 9.3.0.

```
<?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 = "">
<add key="SSLKeyRepository" value="" />
<add key="SSLCipherSpec" value="" />
<add key="SSLPeerName" value="" />
<add key="SSLKeyResetCount" value="" />
<add key="SSLCertRevocationCheck" value="" />
<add key="CertificateLabel" value="" />
<add key="OutboundSNI" value="" />
</dnetxa>
</dnetxa>
</configuration>
```

WmqDotNetXAMonitor
Aplikace Monitor Application vytvoří v adresáři aplikace soubor protokolu pro protokolování průběhu monitorování a stavu zotavení transakce. Protokolování začíná názvem připojení a podrobnostmi kanálu pro zobrazení aktuálního správce front, pro kterého je zotavení spuštěno.

Po spuštění zotavení se zaprotokoluje MessageId zprávy o zotavení transakce, TransactionId neúplné transakce a skutečný výsledek transakce podle koordinace správce transakcí.

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
```

Psaní a nasazení programů IBM MQ .NET

Chcete-li použít IBM MQ classes for .NET pro přístup k frontám IBM MQ , zapisujete programy v libovolném jazyce podporovaném produktem .NET , který obsahuje volání, která vkládají zprávy do front IBM MQ a z nich zprávy.

Dokumentace k produktu IBM MQ obsahuje pouze informace o jazycích C#, C++ a Visual Basic.

Tato kolekce témat poskytuje informace, které pomáhají při psaní aplikací pro interakci se systémy IBM MQ . Podrobnosti o jednotlivých třídách viz Třídy a rozhraní IBM MQ .NET.

Rozdíly v připojení

Způsob, jakým programujete produkt IBM MQ.NET , závisí na režimech připojení, které chcete použít.

Pokud se produkt IBM MQ classes for .NET používá jako spravovaný klient, existuje řada rozdílů oproti standardnímu produktu IBM MQ MQI client, protože některé funkce nejsou pro spravovaného klienta k dispozici.

IBM MQ.NET určuje typ připojení, který má být použit, na základě nastavení určeného pro název připojení, název kanálu, hodnotu přizpůsobení NMQ_MQ_LIB a vlastnost MQC.TRANSPORT_PROPERTY.

Připojení spravovaných klientů

Pokud se produkt IBM MQ classes for .NET používá jako spravovaný klient, existuje řada rozdílů oproti standardu IBM MQ MQI client.

Pro spravovaného klienta nejsou k dispozici následující funkce:

- Komprese kanálů
- Řetězení uživatelské procedury kanálu

Pokusíte-li se tyto funkce použít se spravovaným klientem, vrátí výjimku MQException. Pokud je chyba zjištěna na konci připojení klienta, použije se kód příčiny MQRC_ENVIRONMENT_ERROR. Pokud je zjištěn na konci serveru, použije se kód příčiny vrácený serverem.

Uživatelské procedury kanálu zapsané pro nespravovaného klienta nefungují. Musíte napsat nové uživatelské procedury speciálně pro spravovaného klienta. Zkontrolujte, zda v tabulce CCDT (Client Channel Definition Table) nejsou zadány žádné neplatné uživatelské procedury kanálu.

Název uživatelské procedury spravovaného kanálu může být dlouhý až 999 znaků. Pokud však k určení názvu uživatelské procedury kanálu použijete CCDT, bude délka omezena na 128 znaků.

Komunikace je podporována pouze přes TCP/IP.

Zastavíte-li správce front pomocí příkazu **endmqm** , může ukončení kanálu připojení serveru ke spravovanému klientu .NET trvat déle než zavření kanálů připojení serveru k jiným klientům.

Pokud jste nastavili parametr *NMQ_MQ_MQ_LIB* na hodnotu `managed` , aby bylo možné používat diagnostiku spravovaného problému 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 klient .NET , který se pokouší připojit ke správci front z/OS , se nezdaří s chybou

MQRC_UOW_ENLISTMENT_ERROR (mqrc=2354) ve volání MQOPEN. Spravovaná aplikace .NET používající transakce XA však bude pracovat s distribuovaným správcem front.

Definování typu připojení, který se má použít

Typ připojení je určen nastavením názvu připojení, názvu kanálu, hodnoty přizpůsobení NMQ_MQ_LIB a vlastnosti MQC.TRANSPORT_PROPERTY.

Název připojení můžete zadat takto:

- Explicitně v 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 v 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 v položce hašovací tabulky v konstruktoru MQQueueManager :

```
public MQQueueManager(String queueManagerName, Hashtable properties)
```

- Jako explicitní hodnotu MQEnvironment

```
MQEnvironment.Channel
```

- Nastavením vlastnosti MQC.CHANNEL_PROPERTY v hašovací tabulce MQEnvironment.properties .

Vlastnost přenosu můžete určit následujícím způsobem:

- Nastavením vlastnosti MQC.TRANSPORT_PROPERTY v položce hašovací tabulky v konstruktoru MQQueueManager :

```
public MQQueueManager(String queueManagerName, Hashtable properties)
```

- Nastavením vlastnosti MQC.TRANSPORT_PROPERTY v hašovací tabulce MQEnvironment.properties .

Vyberte požadovaný typ připojení pomocí jedné z následujících hodnot:

MQC.TRANSPORT_MQSERIES_BINDINGS -připojit se jako server
MQC.TRANSPORT_MQSERIES_CLIENT -připojit se jako klient jiný než XA
MQC.TRANSPORT_MQSERIES_XACLIENT -připojte se jako klient XA
MQC.TRANSPORT_MQSERIES_MANAGED -připojit se jako spravovaný klient bez podpory XA

Můžete nastavit hodnotu přizpůsobení NMQ_MQ_LIB tak, aby explicitně zvolila typ připojení, jak je uvedeno v následující tabulce.

Hodnota NMQ_MQ_MQ_LIB	**Typ připojení**
mqic.dll	Připojit se jako klient jiný než XA
mqicxa.dll	Připojit se jako klient XA
mqm.dll	Připojit se jako server nebo jako klient jiný než XA
Spravováno	Připojit se jako spravovaný klient bez podpory XA

Poznámka: Hodnoty mqic32.dll a mqic32xa.dll jsou přijaty jako synonyma mqic.dll a mqicxa.dll pro kompatibilitu s dřívějšími verzemi. Soubory mqm.dll a mqm.pdb jsou však pouze součástí balíku klienta počínaje produktem IBM WebSphere MQ 7.1 .

Pokud zvolíte typ připojení, který je ve vašem prostředí nedostupný, například zadáte mqic32xa.dll a nemáte podporu XA, IBM MQ.NET vygeneruje výjimku.

Nastavení parametru NMQ_MQ_LIB na hodnotu "managed" způsobí, že klient bude používat spravované diagnostické testy problémů IBM MQ , .NET převod dat a další spravované funkce nízké úrovně IBM MQ .

Všechny ostatní hodnoty parametru NMQ_MQ_MQ_LIB způsobují, že proces .NET používá nespravované diagnostické testy a převody dat systému IBM MQ a další nespravované funkce nízké úrovně IBM MQ (za předpokladu, že je v systému nainstalován server IBM MQ MQI client nebo server).

Produkt IBM MQ.NET zvolí typ připojení takto:

1. Je-li MQC.TRANSPORT_PROPERTY , připojuje se podle hodnoty MQC.TRANSPORT_PROPERTY.

   Všimněte si však, že nastavení produktu MQC.TRANSPORT_PROPERTY na MQC.TRANSPORT_MQSERIES_MANAGED nezaručuje, že proces klienta bude spuštěn spravovaným. I s tímto nastavením není klient spravován v následujících případech:

   - Pokud se k produktu MQC.TRANSPORT_PROPERTY nastaveno na jinou hodnotu než MQC.TRANSPORT_MQSERIES_MANAGED.
   - Není-li parametr NMQ_MQ_MQ_LIB nastaven na hodnotu "spravovaný", nejsou testy diagnostiky problémů, převod dat a další funkce nízké úrovně 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 zadán název kanálu bez názvu připojení, dojde k chybě.
3. Pokud byl zadán název připojení i název kanálu:
   - Je-li parametr NMQ_MQ_LIB nastaven na hodnotu mqic32xa.dll, připojí se jako klient XA.
   - Je-li parametr NMQ_MQ_MQ_LIB nastaven na spravovaný, připojí se jako spravovaný klient.
   - Jinak se připojí jako klient jiný než XA.
4. Je-li zadána hodnota NMQ_MQ_LIB, připojí se podle hodnoty NMQ_MQ_MQ_LIB.
5. Pokud je nainstalován server IBM MQ , připojí se jako server.
6. Je-li nainstalován produkt IBM MQ MQI client , připojí se jako klient jiný než XA.
7. Jinak se připojí jako spravovaný klient.

Windows Použití šablony projektu IBM MQ .NET

Klient produktu IBM MQ .NET vám nabízí možnost používat šablonu projektu, která vám pomůže při vývoji vašich aplikací .NET Core .

Než začnete

V systému musíte mít Microsoft Visual Studio 2017nebo novější a .NET Core 2.1 .

Musíte zkopírovat šablonu .NET z

`&MQ_INSTALL_ROOT&\tools\dotnet\samples\cs\core\base\ProjectTemplates\IBMMQ.NETClientApp.zip`

do adresáře

`&USER_HOME_DIRECTORY&\Documents\&Visual_Studio_Version&\Templates\ProjectTemplates`

adresář, kde:

- *& MQ_INSTALL_ROOT* je kořenový adresář vaší instalace.
- *& USER_HOME_DIRECTORY* je váš domovský adresář.

Chcete-li vyzvednout šablonu, musíte zastavit a restartovat produkt Microsoft Visual Studio .

Informace o této úloze

Šablona projektu .NET obsahuje nějaký obecný kód, který můžete použít k usnadnění vývoje aplikací. Pomocí vestavěného kódu se můžete připojit ke správci front IBM MQ a provést operaci vložení nebo získání prostou úpravou vlastností v vestavěném kódu.

Postup

1. Otevřete produkt Microsoft Visual Studio.
2. Klepněte na volbu **Soubor**, následovanou volbou **Nový** a poté volbou **Projekt**.
3. V *okně Vytvořit nový projekt*vyberte volbu `IBM MQ .NET Client App (.NET Core)` a klepněte na tlačítko **Další**.
4. V okně *Konfigurovat nový projekt* změňte *Název projektu* svého projektu, pokud chcete, a klepnutím na tlačítko **Vytvořit** vytvořte projekt .NET .

   `MQDotnetApp.cs` je soubor, který je vytvořen spolu se souborem projektu. Tento soubor obsahuje kód, který se připojuje ke správci front, a provádí operaci vložení a získání.

   Vlastnosti připojení jsou nastaveny na výchozí hodnoty:

   - MQC.CONNECTION_NAME_PROPERTY je nastaveno na *localhost (1414)*
   - MQC.CHANNEL_PROPERTY je nastaveno na *DOTNET.SVRCONN*

   Fronta je nastavena na hodnotu *Q1*a tyto vlastnosti můžete odpovídajícím způsobem upravit.
5. Zkompilujte a spusťte aplikaci.

Související pojmy

Funkce a komponenty IBM MQ

.NET běhové prostředí aplikace- Windows pouze

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 mají prioritu.

Konfigurační soubor klienta

Aplikace klienta IBM MQ classes for .NET může použít konfigurační soubor klienta stejným způsobem jako jakýkoli jiný IBM MQ MQI client. Tento soubor se obvykle nazývá `mqclient.ini`, ale můžete zadat jiný

název souboru. Další informace o konfiguračním souboru klienta viz IBM MQ MQI client configuration file, `mqclient.ini`.

Pouze následující atributy v konfiguračním souboru IBM MQ MQI client jsou důležité pro IBM MQ classes for .NET. Zadáte-li jiné atributy, nebude to mít žádný účinek.

Tabulka 77. Atributy konfiguračního souboru klienta, které jsou relevantní pro IBM MQ classes for .NET

sekce	**Atribut**
kanály	CCSID
kanály	Adresář ChannelDefinition
kanály	Soubor ChannelDefinition
kanály	ReconDelay
kanály	DefRecon
kanály	MQReconnectTimeout
kanály	Parametry ServerConnection
kanály	Put1DefaultAlwaysSync
kanály	PasswordProtection
ClientExitCesta	ExitsDefaultPath
ClientExitCesta	ExitsDefaultPath64
MessageBuffer	MaximumSize
MessageBuffer	PurgeTime
MessageBuffer	UpdatePercentage
V 9.3.0 V 9.3.0 Zabezpečení	Soubor MQIInitialKey
V 9.3.0 V 9.3.0 SSL	SSLKeyRepository
V 9.3.0 V 9.3.0 SSL	Heslo SSLKeyRepository
TCP	ClntRcvBufSize
TCP	ClntSndBufSize
TCP	IPAddressVersion
TCP	KeepAlive

Jakýkoli z těchto atributů můžete přepsat pomocí odpovídající proměnné prostředí.

Konfigurační soubor aplikace

Pokud pracujete s typem spravovaného připojení, můžete také přepsat konfigurační soubor klienta IBM MQ a ekvivalentní proměnné prostředí pomocí konfiguračního souboru aplikace .NET .

Nastavení konfiguračního souboru aplikace .NET jsou prováděna pouze při spuštění se spravovaným typem připojení a jsou ignorována pro ostatní typy připojení.

Konfigurační soubor aplikace .NET a jeho formát jsou definovány produktem Microsoft pro obecné použití v rámci produktu .NET , ale konkrétní názvy sekcí, klíče a hodnoty uvedené v této dokumentaci jsou specifické pro produkt IBM MQ.

Formát konfiguračního souboru aplikace .NET je počet *sekcí*. 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 k řízení vlastnosti KeepAlive TCP/IP:

```
<configuration>
  <configSections>
    <section name="TCP" type="System.Configuration.NameValueSectionHandler"/>
  </configSections>
  <TCP>
    <add key="KeepAlive" value="true"></add>
  </TCP>
<configuration>
```

Klíčová slova použitá v názvech a klíčích sekce konfiguračního souboru aplikace .NET přesně odpovídají klíčovým slovům pro sekce a atributy definované v konfiguračním souboru klienta.

Sekce `<configSections>` musí být prvním podřízeným prvkem prvku `<configuration>` .

Další informace naleznete v dokumentaci k produktu Microsoft .

Příklad fragmentu kódu C# pro použití s .NET

Fragment kódu C#, který demonstruje, že se aplikace připojuje ke správci front, vkládá zprávu do fronty a přijímá odpověď.

Následující fragment kódu C# demonstruje aplikaci, která provádí tři akce:

1. Připojit se ke správci front
2. Vložte zprávu do SYSTEM.DEFAULT.LOCAL.QUEUE
3. Získat zprávu zpět

Také ukazuje, jak změnit typ připojení.

```
// ============================================================================
// Licensed Materials - Property of IBM
// 5724-H72
// (c) Copyright IBM Corp. 2003, 2024
// ============================================================================
using System;
using System.Collections;

using IBM.WMQ;

class MQSample
{
  // The type of connection to use, this can be:-
  // MQC.TRANSPORT_MQSERIES_BINDINGS for a server connection.
  // MQC.TRANSPORT_MQSERIES_CLIENT for a non-XA client connection
  // MQC.TRANSPORT_MQSERIES_XACLIENT for an XA client connection
  // MQC.TRANSPORT_MQSERIES_MANAGED for a managed client connection
  const String connectionType = MQC.TRANSPORT_MQSERIES_CLIENT;

  // Define the name of the queue manager to use (applies to all connections)
  const String qManager = "your_Q_manager";

  // Define the name of your host connection (applies to client connections only)
  const String hostName = "your_hostname";

  // Define the name of the channel to use (applies to client connections only)
  const String channel = "your_channelname";



  /// <summary>
  /// Initialise the connection properties for the connection type requested
  /// </summary>
  /// <param name="connectionType">One of the MQC.TRANSPORT_MQSERIES_ values</param>
  static Hashtable init(String connectionType)
  {
    Hashtable connectionProperties = new Hashtable();

    // Add the connection type
```

```
    connectionProperties.Add(MQC.TRANSPORT_PROPERTY, connectionType);

    // Set up the rest of the connection properties, based on the
    // connection type requested
    switch(connectionType)
    {
      case MQC.TRANSPORT_MQSERIES_BINDINGS:
        break;
      case MQC.TRANSPORT_MQSERIES_CLIENT:
      case MQC.TRANSPORT_MQSERIES_XACLIENT:
      case MQC.TRANSPORT_MQSERIES_MANAGED:
        connectionProperties.Add(MQC.HOST_NAME_PROPERTY, hostName);
        connectionProperties.Add(MQC.CHANNEL_PROPERTY, channel);
        break;
    }

    return connectionProperties;
  }
  /// <summary>
  /// The main entry point for the application.
  /// </summary>
  [STAThread]
  static int Main(string[] args)
  {
    try
    {
      Hashtable connectionProperties = init(connectionType);

      // Create a connection to the queue manager using the connection
      // properties just defined
      MQQueueManager qMgr = new MQQueueManager(qManager, connectionProperties);

      // Set up the options on the queue we want to open
      int openOptions = MQC.MQOO_INPUT_AS_Q_DEF | MQC.MQOO_OUTPUT;

      // Now specify the queue that we want to open,and the open options
      MQQueue system_default_local_queue =
        qMgr.AccessQueue("SYSTEM.DEFAULT.LOCAL.QUEUE", openOptions);

      // Define 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 IBM MQ

Před použitím připojení klienta pro připojení ke správci front je třeba nastavit prostředí IBM MQ .

Poznámka: Tento krok není nezbytný při použití IBM MQ classes for .NET v režimu vazeb serveru.

Programovací rozhraní .NET vám 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, například ty, které jsou uvedeny v následujícím seznamu:

- Název kanálu
- Název hostitele
- Číslo portu
- Uživatelské procedury kanálu
- Parametry zabezpečení SSL
- ID uživatele a heslo

Úplné informace o třídě MQEnvironment viz MQEnvironment.NET class

Chcete-li zadat název kanálu a název hostitele, použijte následující kód:

```
MQEnvironment.Hostname = "host.domain.com";
MQEnvironment.Channel  = "client.channel";
```

Standardně se klienti pokusí připojit k modulu listener IBM MQ na portu 1414. Chcete-li zadat jiný port, použijte kód:

```
MQEnvironment.Port = nnnn;
```

Připojení ke správci front a odpojení od něj

Po konfiguraci prostředí 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, zavolejte na správci front metodu `Disconnect` :

```
queueManager.Disconnect();
```

Při pokusu o připojení ke správci front musíte mít oprávnění dotazovat se (`inq`) na správce front. Bez oprávnění k dotazování se pokus o připojení nezdaří.

Pokud zavoláte metodu `Disconnect` , všechny otevřené fronty a procesy, ke kterým jste přistoupili prostřednictvím tohoto správce front, se zavřou. Je však dobrým zvykem tyto prostředky explicitně zavřít, až je dokončíte používat. Chcete-li zavřít prostředky, použijte metodu `Close` na objektu přidruženém ke každému prostředku.

Metody `Commit` a `Backout` ve správci front nahrazují volání MQCMIT a MQBACK používaná s procedurálním rozhraním.

Přístup k frontám a tématům

K frontám a tématům lze přistupovat pomocí metod MQQueueManager nebo příslušných konstruktorů.

Pro přístup k frontám použijte metody třídy MQQueueManager . Struktura deskriptoru objektu MQOD (object descriptor structure) je sbalena do parametrů těchto metod. Chcete-li například otevřít frontu ve správci front reprezentovaném 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 ve volání MQOPEN.

Metoda AccessQueue vrací nový objekt třídy MQQueue.

Po dokončení použití fronty ji zavřete pomocí metody Close (), jak je uvedeno v následujícím příkladu:

```
queue.Close();
```

Pomocí produktu IBM MQ .NETmůžete také vytvořit frontu pomocí konstruktoru MQQueue. Parametry jsou přesně stejné jako pro metodu accessQueue s přidáním parametru správce front, který určuje objekt MQQueueManager převedený na instanci, který má být použit. Příklad:

```
MQQueue queue = new MQQueue(queueManager,
                            "qName",
                            MQC.MQOO_OUTPUT,
                            "qMgrName",
                            "dynamicQName",
                            "altUserId");
```

Vytvoření objektu fronty tímto způsobem umožňuje psát vlastní podtřídy MQQueue.

Podobně můžete přistupovat k tématům také pomocí metod třídy MQQueueManager . K otevření tématu použijte metodu AccessTopic(). Vrací nový objekt třídy MQTopic. Po dokončení používání tématu jej zavřete pomocí metody Close () objektu MQTopic.

Můžete také vytvořit téma pomocí konstruktoru MQTopic. Pro témata existuje řada konstruktorů; další informace viz `MQTopic`.NET třída.

Zpracování zpráv

Se zprávami se zachází pomocí metod tříd front nebo témat. Chcete-li sestavit novou zprávu, vytvořte nový objekt MQMessageObject.

Vkládání zpráv do front nebo témat pomocí metody Put () třídy MQQueue nebo MQTopic. Získejte zprávy z front nebo témat pomocí metody Get () třídy MQQueue nebo MQTopic. Na rozdíl od procedurálního rozhraní, kde MQPUT a MQGET vkládají a dostávají pole bajtů, IBM MQ classes for .NET vkládají a vkládají instance třídy MQMessage. Třída MQMessage zapouzdřuje datovou vyrovnávací paměť, která obsahuje skutečná data zprávy, spolu se všemi parametry MQMD (deskriptor zprávy), které popisují tuto zprávu.

Chcete-li sestavit novou zprávu, vytvořte novou instanci třídy MQMessage a pomocí metod WriteXXX vložte data do vyrovnávací paměti zpráv.

Při vytvoření nové instance zprávy jsou všechny parametry MQMD automaticky nastaveny na výchozí hodnoty, jak je definováno v poli Počáteční hodnoty a deklarace jazyka pro MQMD. Metoda Put ()

fronty MQQueue rovněž používá jako parametr instanci třídy voleb MQPutMessage. 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 () fronty MQQueue vrací novou instanci MQMessage, která představuje zprávu právě převzatou z fronty. Jako parametr bere také instanci třídy voleb MQGetMessage. Tato třída představuje strukturu MQGMO.

Není třeba zadávat maximální velikost zprávy, protože metoda Get () automaticky upraví velikost vnitřní vyrovnávací paměti tak, aby odpovídala příchozí zprávě. Pro přístup 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();
String name1 = theMessage.ReadUTF();
```

Formát čísel, který používají metody čtení a zápisu, můžete změnit nastavením členské proměnné *kódování* .

Znakovou sadu, která se má použít pro čtení a zápis řetězců, můžete změnit nastavením členské proměnné *characterSet* .

Další podrobnosti viz MQMessage.NET třída .

Poznámka: Metoda WriteUTF() MQMessage automaticky kóduje délku řetězce a také bajty Unicode, které obsahuje. Když bude vaše zpráva přečtena jiným programem .NET (pomocí ReadUTF()), jedná se o nejjednodušší způsob, jak odeslat informace o řetězci.

Zpracování vlastností zprávy

Vlastnosti zprávy vám umožňují vybrat zprávy nebo načíst informace o zprávě bez přístupu k jejím záhlavím. Třída MQMessage obsahuje metody pro získání a nastavení vlastností.

Pomocí vlastností zprávy můžete povolit aplikaci vybrat zprávy ke zpracování nebo načíst informace o zprávě bez přístupu k záhlavím MQMD nebo MQRFH2 . Také usnadňují komunikaci mezi aplikacemi IBM MQ a JMS . Další informace o vlastnostech zprávy v části IBM MQnaleznete v tématu Vlastnosti zprávy.

Třída MQMessage poskytuje řadu metod pro získání a nastavení vlastností podle datového typu vlastnosti. Metody get mají názvy ve formátu Get * Property a metody set mají názvy ve formátu Set * Property, kde hvězdička (*) představuje jeden z následujících řetězců:

- Logická hodnota
- bajt
- Bajty
- Dvojitý
- 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 dokončí.

Ošetření chyb

Ošetřit chyby vzniklé v důsledku IBM MQ classes for .NET použití bloků `try` a `catch` .

Metody v rozhraní .NET nevracejí kód dokončení a kód příčiny. Namísto toho vygenerují výjimku vždy, když kód dokončení a kód příčiny, který je výsledkem volání IBM MQ , nejsou obě nula. To zjednodušuje logiku programu, abyste nemuseli kontrolovat návratové kódy po každém volání IBM MQ. Můžete se rozhodnout, na kterých místech ve vašem programu se chcete vypořádat s možností selhání. V těchto bodech můžete kód obklopit bloky `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 );
}
```

Získávání a nastavení hodnot atributů

Třídy MQManagedObject, MQQueue a MQQueueManager obsahují metody, které umožňují získat a nastavit jejich hodnoty atributů. Mějte na zřeteli, že pro frontu MQQueue metody fungují pouze v případě, že při otevření fronty zadáte příslušné příznaky inquire a Set.

Pro běžné atributy dědí třídy MQQueueManager a MQQueue třídu s názvem MQManagedObject. Tato třída definuje rozhraní Inquire () a Set ().

Když vytvoříte nový objekt správce front pomocí operátoru *new* , automaticky se otevře pro zjišťování. Když použijete metodu AccessQueue() pro přístup k objektu fronty, tento objekt se automaticky neotevře pro operace dotazu nebo nastavení, může to způsobit problémy s některými typy vzdálených front. Chcete-li použít metody Inquire a Set a nastavit vlastnosti ve frontě, musíte zadat odpovídající příznaky dotazu a nastavení v parametru openOptions metody AccessQueue().

Metody inquire a set používají tři parametry:

- pole selektorů
- Pole intAttrs
- Pole charAttrs

Nepotřebujete parametry SelectorCount, IntAttrCount a CharAttrLength, které se nacházejí v MQINQ, protože délka pole je vždy známa. 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 lze zjišťovat. Podmnožina atributů je vystavena jako vlastnosti objektu. Seznam atributů objektů viz Atributy objektů. Informace o vlastnostech objektu naleznete v příslušném popisu třídy.

Vícevláknové programy

Běhové prostředí .NET je ze své podstaty vícevláknové. Produkt IBM MQ classes for .NET umožňuje sdílení objektu správce front ve více podprocesech, ale zajišťuje synchronizaci veškerého přístupu 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 dojde k inicializaci aplikace v jednom podprocesu a kód, který se provádí jako odpověď na tlačítko, se spustí 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 takové, že podproces, který chce vydat volání do správce front, je blokován, dokud nebudou všechna ostatní probíhající volání pro dané připojení dokončena. Pokud vyžadujete souběžný přístup ke stejnému správci front z více podprocesů v rámci programu, vytvořte nový objekt MQQueueManager pro každý podproces, který vyžaduje souběžný přístup. (Jedná se o ekvivalent k zadání samostatného volání MQCONN pro každý podproces.)

Pokud jsou výchozí volby připojení potlačeny produktem MQC.MQCNO_HANDLE_SHARE_NONE nebo MQC.MQCNO_SHARE_NO_BLOCK , správce front již není synchronizován.

Použití tabulky definic kanálů klienta s volbou .NET

Tabulku CCDT (Client Channel Definition Table) můžete použít s tabulkou IBM MQ classes for .NET. Umístění tabulky CCDT určíte různými způsoby v závislosti na tom, zda používáte spravované nebo nespravované připojení.

Typ připojení nespravovaného klienta bez podpory XA nebo XA

S typem nespravovaného připojení můžete určit umístění tabulky CCDT dvěma způsoby:

- Pomocí proměnných prostředí MQCHLLIB zadejte adresář, ve kterém je tabulka umístěna, a pomocí MQCHLTAB zadejte název souboru tabulky.
- Použití konfiguračního souboru klienta. V sekci CHANNELS použijte atributy **ChannelDefinitionDirectory** k uvedení adresáře, kde je tabulka umístěna, a **ChannelDefinitionFile** k uvedení názvu souboru.

Je-li umístění zadáno jak v konfiguračním souboru klienta, tak pomocí proměnných prostředí, mají proměnné prostředí prioritu. Pomocí této funkce můžete určit standardní umístění v konfiguračním souboru klienta a v případě potřeby ji přepsat pomocí proměnných prostředí.

Typ připojení spravovaného klienta

Pomocí typu spravovaného připojení můžete určit umístění tabulky CCDT třemi způsoby:

- Použití konfiguračního souboru aplikace .NET . V sekci CHANNELS použijte klíče **ChannelDefinitionDirectory** k určení adresáře, kde je tabulka umístěna, a **ChannelDefinitionFile** k určení názvu souboru.
- Pomocí proměnných prostředí MQCHLLIB zadejte adresář, ve kterém je tabulka umístěna, a pomocí MQCHLTAB zadejte název souboru tabulky.
- Použití konfiguračního souboru klienta. V sekci CHANNELS použijte atributy **ChannelDefinitionDirectory** k uvedení adresáře, kde je tabulka umístěna, a **ChannelDefinitionFile** k uvedení názvu souboru.

Je-li umístění určeno více než jedním z těchto způsobů, mají proměnné prostředí přednost před konfiguračním souborem klienta a konfigurační soubor aplikace .NET má přednost před oběma jinými metodami. Pomocí této funkce můžete určit standardní umístění v konfiguračním souboru klienta a v případě potřeby jej přepsat pomocí proměnných prostředí nebo konfiguračního souboru aplikace.

Před produktem IBM MQ 9.3.0existuje rozdíl v chování mezi spravovaným klientem .NET a klienty IBM MQ Java a C při použití tabulky CCDT se seskupením správců front. Pokud soubor CCDT obsahuje skupinu správců front tří správců front a tři explicitní příkazy CLNTCONNs pro stejné tři správce front a aplikace poskytuje "*" jako správce front, klienti C a Java vrátí hodnotu MQRC_Q_MGR_NAME_ERROR. Spravovaný klient .NET však používá první dostupný CLNTCONN a pokud žádný není k dispozici, použije správce front seskupeného CLNTCONN.

V 9.3.0 V 9.3.0 V produktu IBM MQ 9.3.0se klient .NET chová stejně jako klienti C a Java a vrací hodnotu MQRC_Q_MGR_NAME_ERROR při použití tabulky CCDT se seskupením správců front.

Jak aplikace .NET určuje, jakou definici kanálu použít

V prostředí klienta IBM MQ .NET lze definici kanálu, která má být použita, zadat různými způsoby. Může existovat více specifikací definice kanálu. Aplikace odvozuje definici kanálu z jednoho nebo více zdrojů.

Pokud existuje více než jedna definice kanálu, použije se ta, která je vybrána v následujícím pořadí priority:

1. Vlastnosti určené v konstruktoru MQQueueManager , a to buď explicitně, nebo pomocí zahrnutí produktu *MQC.CHANNEL_PROPERTY* v hašovací tabulce vlastností
2. Vlastnost *MQC.CHANNEL_PROPERTY* v hašovací tabulce MQEnvironment.properties
3. Vlastnost *Channel* v prostředí MQEnvironment
4. Konfigurační soubor aplikace .NET , název sekce CHANNELS, klíč ServerConnectionParms (platí pouze pro spravovaná připojení)
5. Proměnná prostředí *MQSERVER*
6. Konfigurační soubor klienta, sekce CHANNELS, parametry atributu ServerConnection
7. Tabulka CCDT (Client Channel Definition Table). Umístění tabulky CCDT je určeno v konfiguračním souboru aplikace .NET (platí pouze pro spravovaná připojení).
8. Tabulka CCDT (Client Channel Definition Table). Umístění tabulky CCDT je určeno pomocí proměnných prostředí *MQCHLIB* a *MQCHLTAB* .
9. Tabulka CCDT (Client Channel Definition Table). Umístění tabulky CCDT je určeno pomocí konfiguračního souboru klienta.

Pro položky 1-3 je definice kanálu sestavena 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 priorit:

1. Hodnota *connName* v konstruktoru MQQueueManager
2. Hodnoty vlastností z hašovací tabulky vlastností MQQueueManager
3. Hodnoty vlastností z hašovací tabulky MQEnvironment.properties
4. Hodnoty nastavené jako pole MQEnvironment (například MQEnvironment.Hostname, MQEnvironment.Port)

Pro položky 4-6 je jako hodnota zadána celá definice kanálu. Nespecifikovaná pole v definici kanálu se použijí jako výchozí nastavení systému. S těmito specifikacemi nejsou sloučeny žádné hodnoty z jiných metod definování kanálů a jejich polí.

Pro položky 7-9 je celá definice kanálu převzata z tabulky CCDT. Pole, která nebyla výslovně zadána při definování kanálu, mají výchozí hodnoty systému. S těmito specifikacemi nejsou sloučeny žádné hodnoty z jiných metod definování kanálů a jejich polí.

Použití uživatelských procedur kanálu 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

Používáte-li 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

Pokud použijete spravované připojení k implementaci uživatelské procedury, definujete novou třídu .NET , která implementuje příslušné rozhraní. V balíku IBM MQ jsou definována tři rozhraní ukončení:

- MQSendExit
- MQReceiveExit
- MQSecurityExit

Poznámka: Uživatelské procedury zapsané pomocí těchto rozhraní nejsou podporovány jako uživatelské procedury kanálu v nespravovaném prostředí.

Následující ukázka definuje třídu, která implementuje všechny tři položky:

```
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 instance objektu MQChannelExit a instance objektu MQChannelDefinition . Tyto objekty představují struktury MQCXP a MQCD definované v procedurálním rozhraní.

Data, která mají být odeslána uživatelskou procedurou pro odesílání, a data přijatá v uživatelské proceduře pro zabezpečení nebo příjem jsou uvedena pomocí parametrů uživatelské procedury.

Při vstupu jsou data s offsetem *dataOffset* s délkou *dataLength* v bajtovém poli *dataBuffer* data, která se chystáte odeslat uživatelskou procedurou pro odesílání, a data přijatá v rámci uživatelské procedury pro příjem nebo zabezpečení. Parametr *dataMaxLength* udává maximální délku (z *dataOffset*) k dispozici pro ukončení v *dataBuffer*. Pozn.: Pro uživatelskou proceduru pro zabezpečení zprávy je možné, aby měla dataBuffer hodnotu null, pokud je to poprvé, kdy je volána uživatelská procedura, nebo když je vybrán konec partnera, aby neodesílal žádná data.

Při návratu by měla být hodnota *dataOffset* a *dataLength* nastavena tak, aby ukazovala na offset a délku v rámci vráceného bajtového pole, které by pak měly používat třídy .NET . V případě uživatelské procedury pro odesílání to označuje data, která by měla odeslat, a v případě uživatelské procedury pro zabezpečení nebo příjem data, která by měla být interpretována. Uživatelská procedura by měla normálně vracet bajtové pole; výjimky jsou uživatelské procedury zabezpečení, které by se mohly rozhodnout neodesílat žádná data, a jakákoli uživatelská procedura volaná s příčinami INIT nebo TERM. Nejjednodušší formou uživatelské procedury, kterou lze zapsat, je tedy ta, která nedělá nic jiného než vrací dataBuffer:

Nejjednodušší možné výstupní těleso je:

```
{
  return dataBuffer;
}
```

Třída MQChannelDefinition

ID uživatele a heslo určené pro spravovanou klientskou aplikaci .NET jsou nastaveny ve třídě IBM MQ .NET MQChannelDefinition , která je předána uživatelské proceduře zabezpečení klienta. Uživatelská procedura pro zabezpečení zprávy zkopíruje ID uživatele a heslo do MQCD.RemoteUserIdentifier a MQCD.RemotePassword (viz “Zápis uživatelské procedury zabezpečení” na stránce 941).

Určení uživatelských procedur kanálu (spravovaný klient)

Pokud při vytváření objektu MQQueueManager (v prostředí MQEnvironment nebo v konstruktoru MQQueueManager) zadáte název kanálu a název připojení, můžete zadat uživatelské procedury kanálu dvěma způsoby.

V pořadí podle pořadí jsou následující:

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 .

Pokud nezadáte název kanálu a název připojení, uživatelské procedury kanálu, které se mají použít, pocházejí z definice kanálu získané z tabulky CCDT (Client Channel Definition Table). Hodnoty uložené v definici kanálu nelze přepsat. Další informace o tabulkách definic kanálů viz Tabulka definic kanálů klienta a “Použití tabulky definic kanálů klienta s volbou .NET” na stránce 574 .

V každém případě má specifikace tvar řetězce s následujícím formátem:

```
Assembly_name(Class_name)
```

Název třídy je úplný název, včetně specifikace oboru názvů, třídy .NET , která implementuje IBM.WMQ.MQSecurityExit, IBM.WMQ.MQSendExit nebo IBM.WMQ.MQReceiveExit (podle potřeby). *název_sestavení* je úplné umístění, včetně přípony souboru, sestavení, ve kterém je umístěna třída. Délka

řetězce je omezena na 999 znaků, pokud používáte vlastnosti MQEnvironment nebo MQQueueManager. Pokud je však název uživatelské procedury kanálu uveden v tabulce CCDT, je omezen na 128 znaků. V případě potřeby kód klienta .NET načte a vytvoří instanci určené třídy pomocí analýzy specifikace řetězce.

Určení uživatelských dat uživatelské procedury kanálu (spravovaný klient)

Uživatelské procedury kanálu mohou mít přidružena uživatelská data. Pokud při vytváření objektu MQQueueManager (v prostředí MQEnvironment nebo v konstruktoru MQQueueManager) zadáte název kanálu a název připojení, můžete zadat uživatelská data dvěma způsoby.

V pořadí podle pořadí jsou následující:

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 MQEnvironment SecurityUser, SendUserData nebo ReceiveUserData.

Pokud nezadáte název kanálu a název připojení, hodnoty uživatelských dat uživatelské procedury, které se mají použít, pocházejí z definice kanálu získané z tabulky CCDT (Client Channel Definition Table). Hodnoty uložené v definici kanálu nelze přepsat. Další informace o tabulkách definic kanálů viz Tabulka definic kanálů klienta a “Použití tabulky definic kanálů klienta s volbou .NET” na stránce 574 .

V každém případě je specifikace řetězec, omezený na 32 znaků.

Automatické opětovné připojení klienta v produktu .NET

Můžete provést automatické opětovné připojení klienta ke správci front během neočekávaného přerušení připojení.

Klient může být neočekávaně odpojen od správce front, pokud se například správce front zastaví nebo dojde k selhání sítě či serveru.

Pokud nedojde k automatickému opětovnému připojení klienta, dojde při selhání připojení k chybě. Můžete použít kód chyby, který vám pomůže znovu ustanovit připojení.

Klient, který používá prostředek automatického opětovného připojení klienta, se nazývá klient s možností opětovného připojení. Chcete-li vytvořit klienta s možností opětovného připojení, zadejte při připojování ke správci front určité volby, které se nazývají volby opětovného připojení.

Pokud je klientská aplikace klientem produktu IBM MQ .NET , může se rozhodnout získat automatické opětovné připojení klienta zadáním příslušné hodnoty pro vlastnost CONNECT_OPTIONS_PROPERTY při vytváření správce front pomocí třídy `MQQueueManager` . Podrobnosti o hodnotách parametru CONNECT_OPTIONS_PROPERTY viz Volby opětovného připojení .

Můžete vybrat, zda se aplikace klienta vždy připojí a znovu připojí ke správci front se stejným názvem, ke stejnému správci front nebo k libovolné sadě správců front, které jsou definovány se stejným názvem `QMNAME` v tabulce připojení klienta (podrobnosti viz Skupiny správců front v tabulce CCDT).

Podpora protokolu TLS (Transport Layer Security) pro produkt .NET

Klientské aplikace IBM MQ classes for .NET podporují šifrování TLS (Transport Layer Security). Protokol TLS poskytuje zabezpečení komunikace přes internet a umožňuje aplikacím typu klient/server komunikovat způsobem, který je důvěrný a spolehlivý.

Související pojmy

Podpora zabezpečení TLS spravovaného klienta IBM MQ.NET

Šifrovací bezpečnostní protokoly: TLS

Podpora TLS pro nespravovaného klienta .NET

Podpora TLS pro nespravovaného klienta .NET je založena na MQI jazyka C a IBM Global Security Kit (GSKit). Produkt C MQI zpracovává operace TLS a produkt GSKit implementuje protokoly zabezpečeného soketu TLS.

Povolení TLS pro nespravovaného klienta .NET
Protokol TLS je podporován pouze pro připojení klienta. Chcete-li povolit protokol TLS, musíte zadat CipherSpec , která se má použít při komunikaci se správcem front, a ta se musí shodovat se specifikací CipherSpec nastavenou v cílovém kanálu.

Chcete-li povolit protokol TLS, zadejte specifikaci CipherSpec pomocí statické členské proměnné SSLCipherSpec prostředí MQEnvironment. Následující příklad se připojí ke kanálu SVRCONN s názvem SECURE.SVRCONN.CHANNEL, který byl nastaven tak, aby vyžadoval TLS s CipherSpec TLS_RSA_WITH_AES_128_CBC_SHA:

V 9.3.0 V 9.3.0

```
MQEnvironment.Hostname         = "your_hostname";
MQEnvironment.Channel          = "SECURE.SVRCONN.CHANNEL";
MQEnvironment.SSLCipherSpec    = "TLS_RSA_WITH_AES_128_CBC_SHA256";
MQEnvironment.SSLKeyRepository = "C:\mqm\key.kdb";
MQQueueManager qmgr = new MQQueueManager("your_Q_manager");
```

Seznam CipherSpecsnaleznete v tématu Určení CipherSpecs .

Vlastnost SSLCipherSpec lze také nastavit pomocí parametru MQC.SSL_CIPHER_SPEC_PROPERTY v hašovací tabulce vlastností připojení.

Chcete-li se úspěšně připojit pomocí protokolu TLS, musí být úložiště klíčů klienta nastaveno pomocí řetězu kořenových certifikátů certifikační autority, ze kterého lze ověřit certifikát předložený správcem front. Podobně, pokud byl parametr SSLClientAuth v kanálu SVRCONN nastaven na hodnotu MQSSL_CLIENT_AUTH_REQUIRED, musí úložiště klíčů klienta obsahovat identifikační osobní certifikát, který je důvěryhodný pro správce front.

Použití rozlišujícího názvu správce front
Správce front se identifikuje pomocí certifikátu TLS, který obsahuje *Rozlišující název* (DN).

Aplikace klienta IBM MQ .NET může toto DN používat, aby se ujistila, že komunikuje se správným správcem front. Vzor DN je uveden pomocí proměnné názvu sslPeerprostředí MQEnvironment. Například nastavení:

```
MQEnvironment.SSLPeerName = "CN=QMGR.*, OU=IBM, OU=WEBSPHERE";
```

umožňuje úspěšné připojení pouze v případě, že správce front předloží certifikát s obecným názvem začínajícím na QMGR., a alespoň dva názvy organizačních jednotek, z nichž první musí být IBM a druhý WEBSPHERE.

Vlastnost SSLPeerName lze také nastavit pomocí parametru MQC.SSL_PEER_NAME_PROPERTY v hašovací tabulce vlastností připojení. Další informace o rozlišujících názvech a pravidlech pro nastavení názvů rovnocenných uzlů naleznete v části Zabezpečení IBM MQ.

Je-li nastaven parametr SSLPeerName , připojení budou úspěšná pouze v případě, že je nastaven na platný vzor a správce front předloží odpovídající certifikát.

Ošetření chyb při použití TLS

Produkt IBM MQ classes for .NET může při připojování ke správci front pomocí protokolu TLS vydat následující kódy příčiny:

MQRC_SSL_NOT_ALLOWED
Vlastnost SSLCipherSpec byla nastavena, bylo však použito připojení vazeb. TLS podporuje pouze připojení klienta.

MQRC_SSL_PEER_NAME_MISMATCH
Vzorek DN uvedený ve vlastnosti SSLPeerName se neshoduje s DN prezentovaným správcem front.

CHYBA-MQRC_SSL_PEER_NAME_ERROR
Vzorek DN uvedený ve vlastnosti SSLPeerName byl neplatný.

V 9.3.0 V 9.3.0 **MQRC_KEY_REPOSITORY_ERROR**
Umístění úložiště klíčů buď není určeno, není platné, nebo k němu nelze přistupovat.

Podpora TLS pro spravovaného klienta .NET

Spravovaný klient .NET používá knihovny produktu Microsoft .NET Framework k implementaci protokolů zabezpečeného soketu TLS. Třída Microsoft System.Net.SecuritySslStream funguje jako proud přes připojené sokety TCP a odesílá a přijímá data přes toto připojení soketu.

Minimální požadovaná úroveň .NET Framework je .NET Framework v3.5. Úroveň podpory šifrovacích algoritmů je založena na úrovni .NET Framework , kterou aplikace používá:

- Pro aplikace založené na .NET Framework úrovních 3.5 a 4.0jsou dostupné protokoly zabezpečeného soketu SSL 3.0 a TSL 1.0.
- Pro aplikace založené na .NET Framework úrovni 4.5jsou dostupné protokoly zabezpečeného soketu SSL 3.0, TLS 1.1 a TLS 1.2.

Možná budete muset přesunout aplikace, které očekávají vyšší podporu protokolu TLS, do novější verze rámce, jak je definováno pro podporu zabezpečení 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 rámci .NET Framework, který aplikace používá. Další informace viz “Podpora protokolu TLS pro spravovaného klienta .NET” na stránce 582.

Podpora CipherSpec
Nastavení TLS pro spravovaného klienta .NET jsou stejné jako pro steamy TLS produktu Microsoft.NET . Další informace naleznete v tématech “Podpora CipherSpec pro spravovaného klienta .NET” na stránce 582 a “Mapování CipherSpec pro spravovaného klienta .NET” na stránce 583.

Úložiště klíčů
Úložiště klíčů na straně klienta je úložiště klíčů Windows . Úložiště na straně serveru je typem úložiště Syntaxe kryptografických zpráv (CMS). Další informace viz “Úložiště klíčů pro spravovaného klienta .NET” na stránce 585.

Certifikáty
Certifikáty TLS podepsané svým držitelem můžete použít k implementaci vzájemného ověření mezi klientem a správcem front. Další informace viz “Použití certifikátů pro spravovaného klienta .NET” na stránce 585.

SSLPEERNAME
V produktu .NETmohou aplikace použít volitelný atribut SSLPEERNAME k určení vzoru rozlišujícího názvu (DN). Další informace viz “SSLPEERNAME” na stránce 586.

shoda s normou FIPS
Programové povolení FIPS není podporováno knihovnou zabezpečení Microsoft.NET . Povolení FIPS je řízeno nastavením zásad skupiny Windows .

Shoda se sadou NSA B
IBM MQ implementuje RFC 6460. Implementace produktu Microsoft.NET pro sadu NSA B je 5430. Toto je podporováno od verze .NET Framework 3.5 dále.

Resetování nebo opětné vyjednávání tajného klíče
Ačkoli třída SSLStream nepodporuje resetování nebo opětovné vyjednávání tajného klíče, pro konzistenci s ostatními klienty IBM MQ umožňuje spravovaný klient .NET nastavit aplikace `SSLKeyResetCount`. Další informace viz “Resetování nebo opětné vyjednávání tajného klíče pro spravovaného klienta .NET” na stránce 587.

Kontrola odvolání
Třída SSLStream podporuje kontrolu odvolání certifikátů, kterou automaticky provádí stroj pro řetězení certifikátů. Další informace viz “Kontrola odvolání” na stránce 587.

Podpora uživatelských procedur zabezpečení systému IBM MQ

Třída SSLStream poskytuje omezenou podporu pro uživatelské procedury zabezpečení systému IBM MQ . Dotazování lokálních a vzdálených certifikátů na získání SSLPeerNamePtr(DN subjektu) a SSLRemCertIssNamePtr (DN vydavatele) je možné, protože to je podporováno v produktu Microsoft.NET. Neexistuje však žádná podpora pro získání atributů, jako jsou DNQ, UNSTRUCTUREDNAME a UNSTRUCTUREDADDRESS, takže tyto hodnoty nelze načíst pomocí uživatelských procedur.

Podpora kryptografického hardwaru

Kryptografický hardware není podporován pro spravovaného klienta .NET .

Podpora pro TLS1.3 na spravovaných IBM MQ .NET a XMS .NET klientech

V 9.3.2

V systémech IBM MQ 9.3.2, IBM MQ .NET a XMS .NET klienti podporují protokol TLS1.3 za předpokladu, že operační systém podporuje protokol TLS1.3.

Spravovaný klient .NET používá knihovny produktu Microsoft .NET Framework k implementaci protokolů zabezpečeného soketu TLS. Třída Microsoft System.Net.SecuritySslStream funguje jako proud přes připojené sokety TCP a odesílá a přijímá data přes toto připojení soketu.

V systému Windowspoužívá systém .NET kanál SCHANNEL a v systému Linux .NET používá pro komunikaci SSL protokol OpenSSL .

Windows Pro IBM MQ .NET klientské aplikace spuštěné na Windows

Produkt Microsoft oznámil, že standardně Windows 11 a Windows Server 2022 podporují šifry TSL1.3 .

Šifrovací sady TLS_AES_128_GCM_SHA256 a TLS_AES_256_GCM_SHA384 jsou standardně povoleny v obou verzích produktu Windows.

Upozornění:

- TLS_CHACHA20_POLY1305_SHA256 Cipher Suite není standardně povolen, ale je podporován.
- Pro klienta IBM MQ .NET s povoleným protokolem TLS1.3 je pro úspěšné připojení ke správci front IBM Global Security Kit (GSKit) verze 8.0.55.29 minimální verze, která je vyžadována na straně správce front.

Linux Pro IBM MQ .NET klientské aplikace spuštěné na Linux

Protože produkt .NET používá OpenSSL v systému Linux pro komunikaci SSL, pro použití TLS1.3je minimální požadavek OpenSSL v1.1.1 .

Navíc, protože produkt .NET používá OpenSSL na systému Linux, všechny šifry podporované OpenSSL by měly fungovat i pro systém .NET .

OpenSSL podporuje následující CipherSpecs pro TLS1.3:

- TLS_AES_256_GCM_SHA384
- TLS_CHACHA20_POLY1305_SHA256
- TLS_AES_128_GCM_SHA256
- TLS_AES_128_CCM_8_SHA256
- TLS_AES_128_CCM_SHA256

Související pojmy

“Mapování CipherSpec pro spravovaného klienta .NET” na stránce 583

Rozhraní IBM MQ.NET udržuje tabulku mapování IBM MQ na Microsoft.NET , která se používá k určení verze protokolu TLS, kterou spravovaný klient potřebuje použít k vytvoření zabezpečeného připojení ke správci front.

Podpora protokolu TLS pro spravovaného klienta .NET
Podpora TLS produktu IBM MQ.NET je založena na třídě SSLStream produktu .NET .

Poznámka: Podpora protokolu TLS pro spravovaného klienta .NET závisí na úrovni rámce .NET , kterou aplikace používá. Další informace viz "Podpora TLS pro spravovaného klienta .NET" na stránce 580.

Aby třída SSLStream produktu Microsoft.NET inicializovala protokol TLS a provedla se správcem front hand-shake, je jedním z požadovaných parametrů, které musíte nastavit, **SSLProtocol**, kde musíte zadat číslo verze TLS, což musí být jedna z následujících hodnot:

- SSL3.0
- TLS1.0
- TLS1.2

Hodnota tohoto parametru je úzce spojena s rodinou protokolů, ke které patří upřednostňovaná CipherSpec . Když SSLStream spustí navázání komunikace TLS se serverem (správcem front), použije verzi TLS uvedenou v části **SSLProtocol** k identifikaci seznamu CipherSpecs , které se mají použít pro vyjednávání.

Produkt IBM MQ.NET nezpřístupňuje žádné vlastnosti, které by aplikace používaly k nastavení této hodnoty. Produkt IBM MQ namísto toho používá tabulku mapování k internímu mapování CipherSpec nastavené na řadu protokolů a identifikuje verzi protokolu SSLProtocol, která se má použít. V této tabulce je zobrazeno mapování jednotlivých podporovaných specifikací CipherSpec mezi produkty Microsoft.NET a IBM MQa verze protokolu, ke které náleží. Další informace viz "Mapování CipherSpec pro spravovaného klienta .NET" na stránce 583.

Podpora CipherSpec pro spravovaného klienta .NET
Nastavení CipherSpec pro aplikaci se používají během navázání komunikace se serverem.

Klienti IBM MQ vám umožňují nastavit hodnotu CipherSpec , která se používá během navázání komunikace se správcem front. Klienti IBM MQ by měli nastavit platnou CipherSpec pro zabezpečené připojení, které má být zavedeno, nejlépe CipherSpec určenou v zásadě skupiny Windows . Ponecháte-li toto pole prázdné, označuje kanál prostého textu bez zabezpečení soketů.

Pro spravovaného klienta IBM MQ.NET jsou nastavení TLS určena pro třídu SSLStream Microsoft.NET . Pro parametr SSLStream, CipherSpecnebo seznam předvoleb CipherSpecslze nastavit pouze v zásadě skupiny Windows , což je nastavení pro celý počítač. SSLStream pak použije uvedenou CipherSpec nebo seznam předvoleb během navázání komunikace se serverem. V případě ostatních klientů systému IBM MQ lze vlastnost CipherSpec nastavit v aplikaci na definici kanálu IBM MQ a stejné nastavení se používá pro vyjednávání TLS. V důsledku tohoto omezení může navázání komunikace TLS vyjednat jakoukoli podporovanou specifikaci CipherSpec bez ohledu na to, co je uvedeno v konfiguraci kanálu IBM MQ . Proto je pravděpodobné, že to bude mít za následek chybu AMQ9631 ve správci front. Chcete-li se vyhnout této chybě, nastavte stejnou specifikaci CipherSpec jako tu, kterou jste nastavili v aplikaci jako konfiguraci TLS v zásadě skupiny Windows .

Nový kód klienta TLS produktu IBM MQ.NET kontroluje pouze, zda byla vyjednána správná verze protokolu. Verze protokolu TLS je odvozena ze specifikace CipherSpec , kterou aplikace nastavuje, a používá se pro navázání komunikace TLS se serverem (správcem front). Proto je při návrhu vyžadováno nastavení CipherSpec v aplikaci spravovaného klienta produktu IBM MQ.NET . Pokud je CipherSpec nastavená klientem IBM MQ jiná než ta z protokolů SSL 3.0, TLS 1.0 a TLS 1.2 , klient IBM MQ spravovaný .NET by standardně vyjednával s jakýmikoli šiframi z protokolů SSL 3.0 nebo TLS 1.0 a nehlásil by chybu.

Poznámka: Pokud hodnota CipherSpec dodaná aplikací není CipherSpec známá pro produkt IBM MQ, klient IBM MQ spravovaný .NET ji ignoruje a vyjednává připojení na základě zásady skupiny systému Windows .

Nastavení CipherSpec

Existují tři způsoby nastavení CipherSpec:

Třída MQEnvironment .NET

Následující příklad uvádí, jak nastavit CipherSpec s 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 specifikaci 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);
```

Windows skupinová zásada

Je-li seznam šifrovacích sad konfigurován prostřednictvím konzoly správy zásad skupiny Windows , musí definice kanálu SVRCONN určovat odpovídající CipherSpec. Odpovídající CipherSpec může být buď generická hodnota, například "ANY_TLS12_OR_HIGHER", nebo specifická hodnota, která se mapuje na nejvyšší sadu Cipher Suite, která by byla vyjednána z uspořádaného seznamu. Použití generických hodnot CipherSpec je doporučeno pro použití s klienty .NET, protože v případě změny pořadí seznamu klientů není nutné měnit konfiguraci SVRCONN CipherSpec .

Použití tabulky CCDT

Produkt IBM MQ.NET podporuje pouze tabulky definic kanálů klienta (soubory .TAB), které se nacházejí v lokálním počítači. Existující soubory CCDT, které mají nastavenou hodnotu CipherSpec , lze použít pro připojení produktu IBM MQ.NET . Hodnota CipherSpec nastavená v kanálu připojení klienta však určuje verzi protokolu TLS a musí také odpovídat sadě CipherSpec v zásadě skupiny Windows .

Související pojmy

“Nastavení prostředí IBM MQ” na stránce 570
Před použitím připojení klienta pro připojení ke správci front je třeba nastavit prostředí IBM MQ .

“Podpora TLS pro spravovaného klienta .NET” na stránce 580
Spravovaný klient .NET používá knihovny produktu Microsoft .NET Framework k implementaci protokolů zabezpečeného soketu TLS. Třída Microsoft System.Net.SecuritySslStream funguje jako proud přes připojené sokety TCP a odesílá a přijímá data přes toto připojení soketu.

Související úlohy

Určení CipherSpecs

Související odkazy

Třída MQEnvironment .NET

Mapování CipherSpec pro spravovaného klienta .NET

Rozhraní IBM MQ.NET udržuje tabulku mapování IBM MQ na Microsoft.NET , která se používá k určení verze protokolu TLS, kterou spravovaný klient potřebuje použít k vytvoření zabezpečeného připojení ke správci front.

Je-li v kanálu SVRCONN určena CipherSpec , pokusí se správce front po dokončení navázání komunikace TLS porovnat tuto položku CipherSpec s vyjednanou specifikací CipherSpec , kterou používá klientská aplikace. Pokud správce front nenalezne odpovídající CipherSpec, komunikace se nezdaří s chybou AMQ9631.

Rozhraní IBM MQ.NET udržuje tabulku mapování IBM MQ na Microsoft.NET CipherSpec . Tato tabulka se používá k určení verze protokolu TLS, kterou chce klient použít k vytvoření zabezpečeného připojení soketu se správcem front. Na základě hodnoty SSLCipherSpec může být verze SSLProtocol TLS 1.0nebo TLS 1.2, v závislosti na používané verzi rámce Microsoft.NET Framework.

Ujistěte se, že jste zadali správnou hodnotu SSLCipherSpec , protože uvedení nesprávné hodnoty může mít za následek použití protokolů SSL 3.0 nebo TLS 1.0 .

Tabulka 78. 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_RSA_AES_128_GCM_SHA256	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	TLS 1.2
ECDHE_RSA_AES_256_GCM_SHA384	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	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

Tabulka 78. Tabulka mapování IBM MQ a Microsoft.NET (pokračování)		
IBM MQ CipherSpec	**Microsoft.NET CipherSpec**	**Verze TLS**
ECDHE_ECDSA_AES_256_GCM_SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384_P521	TLS 1.2
V 9.3.2 TLS_AES_256_GCM_SHA384	TLS_AES_256_GCM_SHA384	TLS 1.3
V 9.3.2 TLS_CHACHA20_POLY1305_SHA256	TLS_CHACHA20_POLY1305_SHA256	TLS 1.3
V 9.3.2 TLS_AES_128_GCM_SHA256	TLS_AES_128_GCM_SHA256	TLS 1.3
V 9.3.2 TLS_AES_128_CCM_8_SHA256	TLS_AES_128_CCM_8_SHA256	TLS 1.3
V 9.3.2 TLS_AES_128_CCM_SHA256	TLS_AES_128_CCM_SHA256	TLS 1.3

Notes:

1. Deprecated Tato CipherSpec TLS_RSA_WITH_3DES_EDE_CBC_SHA je zamítnutá. Přesto jej lze použít k přenosu až 32 GB dat před ukončením připojení s chybou AMQ9288. Chcete-li se vyhnout této chybě, musíte se buď vyhnout použití trojitého DES, nebo povolit reset tajného klíče při použití této CipherSpec.

Související pojmy

“Podpora TLS pro spravovaného klienta .NET” na stránce 580
Spravovaný klient .NET používá knihovny produktu Microsoft .NET Framework k implementaci protokolů zabezpečeného soketu TLS. Třída Microsoft System.Net.SecuritySslStream funguje jako proud přes připojené sokety TCP a odesílá a přijímá data přes toto připojení soketu.

Úložiště klíčů pro spravovaného klienta .NET
Úložiště klíčů používané spravovanými klienty .NET je úložiště klíčů Windows . Certifikáty a soukromé klíče musí být k dispozici v úložišti klíčů uživatele nebo systému, aby je mohla klientská aplikace používat pro identitu i důvěryhodnost během navázání komunikace TLS.

Strana klienta

V aplikaci můžete nastavit jednu z následujících hodnot pro úložiště klíčů:

- "*USER": IBM MQ.NET přistupuje k úložišti certifikátů aktuálního uživatele za účelem načtení certifikátů klienta.
- "*SYSTEM": IBM MQ.NET přistupuje k účtu lokálního počítače za účelem načtení certifikátů.

Certifikáty klienta musí být uloženy v úložišti mých certifikátů uživatele nebo účtu počítače. 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:

- Výměna osobních informací-PKCS #12 (.PFX, .P12)
- Standard syntaxe kryptografických zpráv-certifikáty PKCS #7 (.P7B)
- Microsoft Serializované úložiště certifikátů (.SST)

Použití certifikátů pro spravovaného klienta .NET
V případě certifikátů klienta IBM MQ spravovaný .NET klient přistupuje k úložišti klíčů Windows a načte všechny certifikáty klienta, které se shodují buď s popiskem certifikátu, nebo s řetězcem.

Při výběru certifikátu, který se má použít, klient IBM MQ spravovaný .NET vždy použije první odpovídající certifikát pro navázání komunikace SSLStream TLS.

Odpovídající certifikáty podle popisku certifikátu

Pokud nastavíte popisek certifikátu, IBM MQ spravovaný .NET klient prohledá úložiště certifikátů Windows s daným názvem popisku, aby identifikoval certifikát klienta. Načte všechny odpovídající certifikáty a použije první certifikát na seznamu. Existují dvě možnosti pro nastavení popisku certifikátu:

- Popisek certifikátu lze nastavit na třídě MQEnvironment, která přistupuje k MQEnvironment.CertificateLabel.
- Popisek certifikátu lze také nastavit ve vlastnostech hašovací tabulky, které jsou dodávány jako vstupní parametr s konstruktorem MQQueueManager , jak ukazuje následující příklad.

```
Hashtable properties = new Hashtable();
properties.Add("CertificateLabel", "mycert");
```

Název ("CertificateLabel") a hodnota rozlišuje velikost písmen.

Odpovídající certifikáty podle řetězce

Není-li popisek certifikátu nastaven, bude vyhledán a použit certifikát, který odpovídá řetězci `"ibmwebspheremq"` a aktuálně přihlášenému uživateli (malými písmeny).

Související úlohy

Bezpečné připojení klienta ke správci front

Související odkazy

Třída MQEnvironment .NET

SSLPEERNAME
Atribut SSLPEERNAME se používá ke kontrole rozlišujícího názvu (DN) certifikátu ze správce front typu peer.

V produktu IBM MQ.NETmohou aplikace použít SSLPEERNAME k určení vzoru rozlišujícího názvu, jak ukazuje následující příklad.

```
SSLPEERNAME(CN=QMGR.*, OU=IBM, OU=WEBSPHERE)
```

Stejně jako u ostatních klientů systému IBM MQ je parametr SSLPEERNAME nepovinný.

Není-li hodnota SSLPEERNAME nastavena, spravovaný klient IBM MQ.NET neprovede žádné ověření vzdáleného certifikátu (serveru) a spravovaný klient pouze přijme vzdálený certifikát (/server) tak, jak je.

Způsob, jakým nastavíte SSLPEERNAME, závisí na tom, kterou z nabídek zásobníku IBM MQ používáte.

IBM MQ classes for .NET
K dispozici jsou tři možnosti, jak je uvedeno níže.

1. Nastavte 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 hodnotu SSLPEERNAME v souboru `Hashtable properties` jako pro volbu 2.

XMS .NET
Nastavte název partnera SSL v továrně připojení:

```
ConnectionFactory.SetStringProperty(XMSC.WMQ_SSL_PEER_NAME, value);
```

WCF-zařízení
Do identifikátoru URI zadejte název SslPeerjako pole oddělené středníkem.

Související odkazy
Třída MQEnvironment .NET

Resetování nebo opětné vyjednávání tajného klíče pro spravovaného klienta .NET
Třída SSLStream nepodporuje reset/renegotiation tajného klíče. Chcete-li však být konzistentní s ostatními klienty IBM MQ , IBM MQ spravovaný .NET klient umožňuje aplikacím nastavit **SSLKeyResetCount**.

Po dosažení tohoto limitu se produkt IBM MQ.NET odpojí od správce front a aplikace obdrží oznámení o této výjimce s kódem příčiny MQRC_CONNECTION_BROKEN. Aplikace se mohou rozhodnout zpracovat výjimku a znovu vytvořit připojení nebo povolit volbu MQCNO_RECONNECT pro produkt IBM MQ.NET , aby se automaticky znovu připojil ke správci front.

Povolení prostředku automatického opětovného připojení klienta znamená, že po dosažení počtu resetů klíče jsou všechna existující připojení vypnutá a klient produktu IBM MQ.NET znovu znovu vytvoří všechna připojení znovu. Další informace o automatickém opětovném připojení klienta naleznete v tématu Automatické opětovné připojení klienta.

Související pojmy
Resetování tajných klíčů SSL a TLS

Kontrola odvolání
Třída SSLStream podporuje kontrolu odvolání certifikátů.

Kontrolu odvolání provádí automaticky stroj pro řetězení certifikátů. To platí jak pro protokol OCSP (Online Certificate Status Protocol), tak pro seznamy odvolaných certifikátů (CRL). Třída SSLStream používá odvolání certifikátu, které používá pouze server uvedený v certifikátu, tj. server je diktován samotným certifikátem. Je možné, aby požadavky HTTP rozšíření CDP a OCSP HTTP na server proxy prostřednictvím serveru proxy HTTP .

Způsob, jakým nastavíte kontrolu odvolání, závisí na tom, kterou z nabídek zásobníku IBM MQ používáte.

IBM MQ.NET
Kontrolu odvolání lze nastavit pomocí přístupu k vlastnosti **MQEnvironment.SSLCertRevocationCheck** v souboru třídy `MQEnvironment.cs` .

XMS .NET
Kontrolu odvolání lze nastavit v kontextu vlastnosti továrny připojení, jak ukazuje následující příklad.

```
ConnectionFactory.SetBooleanProperty(XMSC.WMQ_SSL_CERT_REVOCATION_CHECK, true);
```

WCF-zařízení
Kontrolu odvolání lze na identifikátoru URI nastavit pomocí následující konvence pojmenování.

```
"SslCertRevocationCheck=true"
```

Konfigurace protokolu TLS pro spravovaný produkt IBM MQ .NET
Konfigurace TLS pro spravovaný produkt IBM MQ .NET se skládá z vytvoření certifikátů podepsaného, následné konfigurace strany serveru, strany klienta a aplikačního programu.

Informace o této úloze

Chcete-li konfigurovat protokol TLS, musíte nejprve vytvořit příslušné certifikáty podepsaného. Certifikáty podepsaného mohou být buď podepsané sebou samým, nebo certifikáty poskytnuté certifikační autoritou. Ačkoli lze certifikáty podepsané svým držitelem používat ve vývojovém, testovacím nebo předprodukčním systému, nepoužívejte je v produkčním systému. V produkčním 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 dodávaných s produktem IBM MQ :

      Použijte grafické rozhraní **strmqikm** , nebo použijte **runmqckm** nebo **runmqakm** z příkazového řádku. Další informace o použití těchto nástrojů naleznete v tématu Použití **runmqckm**, **runmqakm**a **strmqikm** ke správě digitálních certifikátů.
   b) Chcete-li získat certifikáty pro správce front a klienty od certifikační autority (CA), postupujte podle pokynů v části Získání osobních certifikátů od certifikační autority.
2. Nakonfigurujte stranu serveru.
   a) Konfigurujte TLS ve správci front pomocí IBM Global Security Kit (GSKit), jak je popsáno v tématu Bezpečné připojení klienta ke správci front.
   b) Nastavte atributy TLS kanálu SVRCONN:
      - Nastavte parametr **SSLCAUTH** na hodnotu `"REQUIRED/OPTIONAL"`.
      - Nastavte **SSLCIPH** na odpovídající CipherSpec.

      Další informace viz“Povolení TLS pro nespravovaného klienta .NET” na stránce 579.
3. Nakonfigurujte stranu klienta.
   a) Importujte certifikáty klienta do úložiště certifikátů Windows (pod účtem uživatele/počítače).

      Produkt IBM MQ .NET přistupuje k certifikátům klienta z úložiště certifikátů Windows , proto musíte certifikáty importovat do úložiště certifikátů Windows , abyste navázali připojení zabezpečeného soketu k produktu IBM MQ . Další informace o přístupu k úložišti klíčů Windows a importu certifikátů na straně klienta naleznete v tématu Import nebo export certifikátů a soukromých klíčů.
   b) Zadejte CertificateLabel , jak je popsáno v tématu Bezpečné připojení klienta ke správci front.
   c) V případě potřeby upravte Windows Skupinové zásady tak, aby nastavovaly specifikaci CipherSpec, a poté restartujte počítač, aby se aktualizace zásad skupiny Windows projevily.
4. Nakonfigurujte aplikační program.
   a) Nastavte hodnotu MQEnvironment nebo hodnotu SSLCipherSpec , abyste označili připojení jako zabezpečené připojení.

      Hodnota, kterou uvedete, se použije k identifikaci používaného protokolu (TLS). Sada CipherSpec by měla být jednou ze specifikací CipherSpecs podporované verze protokolu SSLProtocol a může být pokud možno stejná jako ta, která je určena v zásadě skupiny Windows . (Podporovaná verze SSLProtocol závisí na použitém rámci .NET . Verze SSLProtocol může být TLS 1.0nebo TLS 1.2, v závislosti na verzi rámce Microsoft .NET , kterou používáte.)

      Poznámka: Pokud hodnota CipherSpec dodaná aplikací není CipherSpec známá pro produkt IBM MQ, klient IBM MQ spravovaný .NET ji ignoruje a vyjednává připojení na základě zásady skupiny systému Windows .
   b) Nastavte vlastnost SSLKeyRepository na hodnotu `"*SYSTEM"` nebo `"*USER"`.
   c) Volitelné: Nastavte SSLPEERNAME na rozlišující název (DN) certifikátu serveru.
   d) Zadejte CertificateLabel , jak je popsáno v tématu Bezpečné připojení klienta ke správci front.
   e) Nastavte další volitelné parametry, které požadujete, například KeyResetCount, CertificationRevocationCheck a povolte FIPS.

Příklady nastavení protokolu TLS a úložiště klíčů TLS

Pro produkt Base .NETmůžete nastavit protokol TLS a úložiště klíčů TLS prostřednictvím třídy MQEnvironment, jak ukazuje následující příklad:

```
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")
```

Volitelně můžete nastavit protokol TLS a úložiště klíčů TLS zadáním hašovací tabulky jako součást konstruktoru MQQueueManager , jak ukazuje následující příklad.

```
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 zahájení vývoje aplikací TLS spravovaných produktem IBM MQ .NET naleznete v tématu“Psaní jednoduché aplikace” na stránce 589.

Související odkazy

Třída MQEnvironment .NET

Počet KeyReset(MQLONG)

Standard FIPS (Federal Information Processing Standards) pro AIX, Linux, and Windows

Psaní jednoduché aplikace

Rady pro zápis jednoduché IBM MQ spravované .NET aplikace TLS, včetně příkladů nastavení vlastností SSL pro továrny připojení, vytvoření instance správce front, připojení, relace a cíle a odeslání testovací zprávy.

Než začnete

Nejprve musíte nakonfigurovat TLS pro spravovaný produkt IBM MQ.NET , jak je popsáno v tématu “Konfigurace protokolu TLS pro spravovaný produkt IBM MQ .NET” na stránce 587.

Pro konfiguraci aplikačního programu v základním souboru .NETnastavte vlastnosti SSL buď pomocí třídy MQEnvironment, nebo zadáním hašovací tabulky jako součásti konstruktoru MQQueueManager .

Pro konfiguraci aplikačního programu v produktu XMS .NETnastavíte vlastnosti SSL v kontextu vlastností továren připojení.

Postup

1. Nastavte vlastnosti 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 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 uvedeno v následujících příkladech.

   Příklad pro 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 připojení TLS a zda správně funguje.

Konfigurace trasování pro SSLStream
Chcete-li zachytit události trasování a zprávy související se třídou SSLStream, musíte přidat sekci konfigurace pro diagnostiku systému do konfiguračního souboru aplikace.

Informace o této úloze

Poznámka:

Tato úloha platí pouze pro IBM MQ classes for .NET Framework . Konfigurační soubor aplikace není podporován v knihovnách IBM MQ classes for .NET (.NET Standard a .NET 6).

Pokud nepřidáte sekci konfigurace pro diagnostiku systému do konfiguračního souboru aplikace, IBM MQ spravovaný .NET klient nezachytí žádné události, trasování nebo body ladění týkající se TLS a třídy SSLStream.

Poznámka: Spuštění IBM MQ trasování pomocí **strmqtrc** nezachytí všechna požadovaná trasování TLS.

Postup

1. Vytvořte soubor konfigurace aplikace (App.Config) pro projekt aplikace.
2. Přidejte sekci konfigurace diagnostiky systému, jak ukazuje následující příklad.

```
<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 jakákoli verze použitého souboru .net `amqmdnet.dll` .

Související úlohy

Trasování klientů IBM MQ classes for .NET Framework pomocí konfiguračního souboru aplikace

Ukázkové aplikace pro implementaci TLS ve spravovaném .NET
K dispozici jsou ukázkové aplikace pro zobrazení implementace protokolu TLS pro spravovaný produkt .NET ve vlastních kanálech IBM MQ classes for .NET, XMS .NET a IBM MQ pro WCF.

Následující tabulka zobrazuje umístění ukázkových aplikací. *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Tabulka 79. Umístění ukázkových aplikací pro implementaci TLS ve spravovaném systému .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

Tabulka 79. Umístění ukázkových aplikací pro implementaci TLS ve spravovaném systému .NET (pokračování)	
Nabídka zásobníku produktu IBM MQ.NET	**Umístění vzorků**
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
IBM MQ vlastní kanál pro WCF	*MQ_INSTALLATION_PATH*\Tools\dotnet\samples\cs\wcf\samples\WCF\oneway\service\MQMessagingOneWayService.cs

Windows Použití monitoru .NET

.NET Monitor je aplikace podobná monitoru spouštěčů IBM MQ .

Důležité: Důležité informace naleznete v části Funkce, které lze použít pouze s primární instalací v systému Windows .

Můžete vytvořit komponenty .NET , které se převedou na instanci, kdykoli je zpráva přijata v monitorované frontě, a které pak zpracují tuto zprávu. Monitor .NET je spuštěn příkazem **runmqdnm** a zastaven příkazem **endmqdnm** . Podrobnosti o těchto příkazech viz runmqdnm a endmqdnm.

Chcete-li použít monitor .NET Monitor, napište komponentu, která implementuje rozhraní IMQObjectTrigger , které je definováno v souboru amqmdnm.dll.

Komponenty mohou být buď transakční, nebo netransakční. Transakční komponenta musí dědit z System.EnterpriseServices.ServicedComponent a být registrována jako RequiresTransaction nebo SupportsTransaction. Nesmí být registrován jako RequiresNew , protože monitor .NET již zahájil transakci.

Komponenta přijímá objekty MQQueueManager, MQQueue a MQMessage z produktu **runmqdnm**. Může také obdržet řetězec parametrů uživatele, pokud byl zadán, pomocí volby příkazového řádku *-u* při spuštění příkazu runmqdnm . Všimněte si, že vaše komponenta obdrží obsah zprávy, která dorazila do monitorované fronty v objektu MQMessage. Nemusí se připojit ke správci front, otevřít frontu nebo získat samotnou zprávu. Komponenta pak musí zprávu zpracovat podle potřeby a vrátit řízení do monitoru .NET Monitor.

Pokud byla vaše komponenta zapsána jako transakční komponenta, zaregistruje se k potvrzení nebo odvolání transakce pomocí zařízení System.EnterpriseServices.ServicedComponent.

Vzhledem k tomu, že komponenta přijímá objekty MQQueueManager a MQQueue a zprávu, obsahuje úplné informace o kontextu pro tuto zprávu a může například otevřít jinou frontu ve stejném správci front bez nutnosti samostatného připojení k produktu IBM MQ.

Windows *Příklady fragmentů kódu*

Toto téma obsahuje dva příklady komponent, které získají zprávu z monitoru .NET Monitor a vytisknou ji, z nichž jedna používá transakční zpracování a druhá netransakční zpracování. Třetí příklad ukazuje běžné obslužné rutiny použitelné pro první dva příklady. Všechny příklady jsou v jazyce C#.

Příklad 1: Transakční zpracování

```
/**************************************************************************/
/* Licensed materials, property of IBM                                    */
/* 63H9336                                                                */
/* (C) Copyright IBM Corp. 2005, 2024.                                    */
/**************************************************************************/
using System;
using System.EnterpriseServices;

using IBM.WMQ;
```

```
using IBM.WMQMonitor;

[assembly: ApplicationName("dnmsamp")]

// build:
//
// csc -target:library -reference:amqmdnet.dll;amqmdnm.dll TranAssembly.cs
//
// run (with dotnet monitor)
//
// runmqdnm -m QMNAME -q QNAME -a dnmsamp.dll -c Tran

namespace dnmsamp
{
  [TransactionAttribute(TransactionOption.Required)]
  public class Tran : ServicedComponent, IMQObjectTrigger
  {
    Util util = null;

    [AutoComplete(true)]
    public void Execute(MQQueueManager qmgr, MQQueue queue,
        MQMessage message, string param)
    {
      util = new Util("Tran");

      if (param != null)
        util.Print("PARAM: '" +param.ToString() + "'");

      util.PrintMessage(message);

      //System.Console.WriteLine("SETTING ABORT");
      //ContextUtil.MyTransactionVote = TransactionVote.Abort;

      System.Console.WriteLine("SETTING COMMIT");
      ContextUtil.SetComplete();
      //ContextUtil.MyTransactionVote = TransactionVote.Commit;
    }
  }
}
```

Příklad 2: Netransakční zpracování

```
/*********************************************************************/
/* Licensed materials, property of IBM                               */
/* 63H9336                                                           */
/* (C) Copyright IBM Corp. 2005, 2024.                               */
/*********************************************************************/

using System;

using IBM.WMQ;
using IBM.WMQMonitor;

// build:
//
// csc -target:library -reference:amqmdnet.dll;amqmdnm.dll NonTranAssembly.cs
//
// run (with dotnet monitor)
//
// runmqdnm -m QMNAME -q QNAME -a dnmsamp.dll -c NonTran
namespace dnmsamp
{
  public class NonTran : IMQObjectTrigger
  {
    Util util = null;

    public void Execute(MQQueueManager qmgr, MQQueue queue,
        MQMessage message, string param)
    {
      util = new Util("NonTran");

      try
      {
        util.PrintMessage(message);
      }

      catch (Exception ex)
      {
```

```
        System.Console.WriteLine(">>> NonTran\n{0}", ex.ToString());
      }
    }
  }
}
```

Příklad 3: Společné rutiny

```
/*********************************************************************/
/* Licensed materials, property of IBM                               */
/* 63H9336                                                           */
/* (C) Copyright IBM Corp. 2005, 2024.                               */
/*********************************************************************/

using System;

using IBM.WMQ;

namespace dnmsamp
{
    /// <summary>
    /// Summary description for Util.
    /// </summary>
    public class Util
    {
    /* ---------------------------------------------------------------- */
    /* Default prefix string of the namespace.                           */
    /* ---------------------------------------------------------------- */
    private string prefixText = "dnmsamp";

    /* ---------------------------------------------------------------- */
    /* Constructor that takes the replacement prefix string to use.     */
    /* ---------------------------------------------------------------- */
    public Util(String text)
    {
      prefixText = text;
    }

    /* ---------------------------------------------------------------- */
    /* Display an arbitrary string to the console.                      */
    /* ---------------------------------------------------------------- */
    public void Print(String text)
    {
      System.Console.WriteLine("{0} {1}\n", prefixText, text);
    }



 /* ---------------------------------------------------------------- */
    /* Display the content of the message passed to the console.        */
    /* ---------------------------------------------------------------- */
    public void PrintMessage(MQMessage message)
    {
      if (message.Format.CompareTo(MQC.MQFMT_STRING) == 0)
      {
        try
        {
          string messageText = message.ReadString(message.MessageLength);

          Print(messageText);
        }

        catch(Exception ex)
        {
          Print(ex.ToString());
        }
      }
      else
      {
        Print("UNRECOGNISED FORMAT");
      }
    }

    /* ---------------------------------------------------------------- */
    /* Convert the byte array into a hex string.                        */
    /* ---------------------------------------------------------------- */
    static public string ToHexString(byte[] byteArray)
    {
```

```
      string hex = "0123456789ABCDEF";

      string retString = "";

      for(int i = 0; i < byteArray.Length; i++)
      {
        int h = (byteArray[i] & 0xF0)>>4;
        int l = (byteArray[i] & 0x0F);

        retString += hex.Substring(h,1) + hex.Substring(l,1);
      }

      return retString;
    }
  }
}
```

Kompilace programů IBM MQ .NET

Vzorové příkazy pro kompilaci aplikací .NET napsaných v různých jazycích.

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

Chcete-li sestavit aplikaci v jazyce 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 Visual Basic 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 aplikaci Managed C++ pomocí IBM MQ classes for .NET, použijte následující příkaz:

```
cl /clr MQ_INSTALLATION_PATH\bin Myprog.cpp
```

Další jazyky naleznete v dokumentaci dodané dodavatelem jazyka.

Použití samostatného klienta IBM MQ .NET

Klient produktu IBM MQ .NET vám nabízí možnost zabalit a implementovat sestavení produktu IBM MQ .NET , aniž byste museli používat úplnou instalaci klienta IBM MQ na produkčních systémech pro spouštění aplikací.

Než začnete

V 9.3.1 V produktu IBM MQ 9.3.1je knihovna klienta `amqmdnetstd.dll` instalovaná ve výchozím umístění založena na .NET 6. Knihovna klienta `amqmdnetstd.dll` založená na .NET Standard byla přesunuta do nového umístění v instalačním balíku klienta IBM MQ a je nyní k dispozici v následujících umístěních:

- Windows V systému Windows: *MQ_INSTALLATION_PATH*`\bin\netstandard2.0`
- Linux V systému Linux: *MQ_INSTALLATION_PATH*`\lib64\netstandard2.0`

LTS Pro produkt IBM MQ 9.3.0 Long Term Supportje knihovna `amqmdnetstd.dll` k dispozici v následujících umístěních:

- Windows V systému Windows: *MQ_INSTALLATION_PATH*`\bin`. Ukázkové aplikace jsou nainstalovány v produktu *MQ_INSTALLATION_PATH*`/samp/dotnet/samples/cs/core/base`.

- Linux V systému Linux: *MQ_INSTALLATION_PATH*/lib64 path. Ukázky .NET jsou v adresáři *MQ_INSTALLATION_PATH*/samp/dotnet/samples/cs/core/base.

Stabilized LTS Knihovna amqmdnet.dll je stále dodávána, ale tato knihovna je stabilizována; to znamená, že do ní nebudou zavedeny žádné nové funkce. Pro všechny nejnovější funkce musíte provést migraci do knihovny amqmdnetstd.dll . Můžete však pokračovat v používání knihovny amqmdnet.dll ve vydáních IBM MQ 9.1 Long Term Support nebo Continuous Delivery .

Informace o této úloze

Aplikace IBM MQ .NET můžete sestavit na počítači, na kterém je nainstalován úplný klient IBM MQ , a později zabalit sestavení IBM MQ .NET , tj. amqmdnetstd.dll, spolu s aplikací a implementovat ji na produkční systémy.

Aplikace, které sestavíte a nasadíte, mohou být tradiční aplikace .NET , služby nebo webové/pracovní aplikace Microsoft Azure .

V takových implementacích klient produktu IBM MQ .NET podporuje pouze spravovaný režim konektivity ke správci front. Vazby serveru a připojení v režimu nespravovaného klienta nejsou k dispozici, protože tyto dva režimy vyžadují úplnou instalaci klienta IBM MQ . Jakýkoli pokus o použití těchto dvou režimů má za následek výjimku aplikace.

Procedura

Odkazování na sestavení klienta IBM MQ .NET v aplikacích

- Odkazujte na sestavení amqmdnetstd.dll ve své aplikaci stejným způsobem, jako jste odkazovali na dřívější vydání.

  Nastavte vlastnost **CopyLocal** sestavení amqmdnetstd.dll na hodnotu True , abyste se ujistili, že je sestavení amqmdnetstd.dll zkopírováno do adresáře bin aplikace. Nastavení této vlastnosti také pomáhá nástroji pro balení aplikací zabalit požadované binární soubory pro implementaci na produkční systémy a také cloudová prostředí Microsoft Azure PaaS .

Přidání podpory globálních transakcí

- Ujistěte se, že aplikace implementuje aplikaci monitoru WMQDotnetXAMonitor na počítač spolu se samotnou aplikací.

  Pokud aplikace používá funkci globálních transakcí spravovaných produktem IBM MQ .NET , musí také implementovat WMQDotnetXAMonitor na počítač spolu se samotnou aplikací. Tento obslužný program je potřebný pro obnovu všech neověřených transakcí.

Spuštění a zastavení trasování

- Chcete-li spustit a zastavit trasování pouze pro systém IBM MQ classes for .NET Framework pomocí konfiguračního souboru aplikace a IBM MQ specifického konfiguračního souboru trasování, viz Trasování tříd IBM MQ pro klienta .NET Framework pomocí konfiguračního souboru aplikace.

  Musíte použít konfigurační soubor aplikace a specifický konfigurační soubor trasování IBM MQ , protože neexistuje žádná úplná instalace klienta IBM MQ , standardní nástroje, které se používají pro spuštění a zastavení trasování, **strmqtrc** a **endmqtrc**, nejsou k dispozici.

  Notes:

  - Tento způsob generování trasování platí pro redistribuovatelného spravovaného klienta .NET i samostatného klienta .NET . Viz .NET běhové prostředí aplikace- Windows pouze.
  - Konfigurační soubor aplikace není podporován v knihovnách IBM MQ classes for .NET (.NET Standard a .NET 6). Chcete-li povolit trasování pro knihovny IBM MQ classes for .NET (.NET Standard a .NET 6), použijte proměnnou prostředí **MQDOTNET_TRACE_ON** . Viz Trasování IBM MQ .NET aplikací používajících proměnné prostředí.

- V 9.3.3

  Spusťte a zastavte trasování pomocí souboru mqclient.ini a nastavte příslušné vlastnosti sekce Trasování.

Viz Trasování IBM MQ .NET aplikací s mqclient.ini.

V produktu IBM MQ 9.3.3můžete konfigurovat trasování pomocí souboru `mqclient.ini` a nastavením příslušných vlastností sekce Trasování. Můžete také dynamicky povolit a zakázat trasování pomocí souboru `mqclient.ini` .

Povolení přesměrování vazby v konfiguračním souboru aplikace

- Chcete-li povolit odkaz na vazbu času kompilace sestavení IBM MQ .NET na novější verzi sestavení, přidejte vlastnost <dependentAssembly> do konfiguračního souboru aplikace.

  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`) sestavení IBM MQ .NET , ale později opravné sady IBM MQ 8.0.0 Fix Pack 3, byla poté použita aktualizovaná sestava IBM MQ.NET 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

“Instalace produktu IBM MQ classes for .NET” na stránce 541
Produkt IBM MQ classes for .NET, včetně ukázek, se instaluje s operačním systému IBM MQ na systémech Windows a Linux .

Redistribuovatelní klienti

.NET běhové prostředí aplikace- Windows pouze

Související úlohy

“Použití aplikace WMQDotnetXAMonitor” na stránce 560
Klient IBM MQ .NET poskytuje aplikaci monitoru XA, WmqDotnetXAMonitor, kterou můžete použít k obnovení nedokončených distribuovaných transakcí. Aplikace WmqDotnetXAMonitor vytvoří připojení ke správci front, kde jsou transakce v nejistém stavu, a poté transakci vyřeší na základě nastavených parametrů.

Trasování IBM MQ aplikací .NET

V 9.3.0 OutboundSNI vlastnost

Vlastnost **OutboundSNI** můžete v aplikaci nastavit buď pomocí vlastnosti, nebo pomocí proměnné prostředí.

V produktu IBM MQ 9.3.0můžete nastavit produkt MQC.OUTBOUND_SNI_PROPERTY v aplikaci s použitím hašovací tabulky při použití třídy `MQQueueManager` pro připojení ke správci front.

MQC.OUTBOUND_SNI_PROPERTY má následující hodnoty:

- MQC.OUTBOUND_SNI_CHANNEL, která se mapuje na "CHANNEL"
- MQC.OUTBOUND_SNI_HOSTNAME, který se mapuje na "HOSTNAME"
- MQC.OUTBOUND_SNI_ASTERISK, která se mapuje na "*"

Dále můžete nastavit vlastnost **OutboundSNI** pomocí proměnné prostředí MQOUTBOUND_SNI, která má následující hodnoty:

- CHANNEL
- HOSTNAME

- *

a nastavte hodnotu **OutboundSNI** v souboru App.config stejně jako u ostatních vlastností mqclient.ini .

Poznámka: Výchozí hodnota vlastnosti je MQC.OUTBOUND_SNI_CHANNEL , pokud není nastavena žádná specifická hodnota.

Pořadí priorit pro nastavení vlastnosti **OutboundSNI** ve spravovaném uzlu je:

1. Vlastnost na úrovni aplikace
2. Proměnná prostředí

Pro vlastnost **OutboundSNI** v nespravovaném uzlu je podporována pouze vlastnost mqclient.ini .

Vlastnosti nastavené v souboru App.config jsou použitelné pouze pro aplikace .NET Framework .

Zadáte-li hodnotu, která není platná na úrovni aplikace nebo v souboru App.config , bude vydán návratový kód MQRC_OUTBOUND_SNI_NOT_VALID.

Pokud nastavíte proměnnou prostředí, která není platná, nebo zadáte hodnotu, která není platná v souboru mqclient.ini , použije se výchozí hodnota CHANNEL .

OutboundSNI a více certifikátů

Produkt IBM MQ používá záhlaví SNI k zajištění funkčnosti více certifikátů. Pokud se aplikace připojuje ke kanálu IBM MQ , který je konfigurován pro použití jiného certifikátu prostřednictvím pole CERTLABL, musí se aplikace připojit s nastavením **OutboundSNI** na hodnotu CHANNEL.

Pokud se aplikace s nastavením **OutboundSNI** na jinou hodnotu než CHANNEL připojí ke kanálu s nakonfigurovaným popiskem certifikátu, aplikace se odmítne s hodnotou MQRC_SSL_INITIALIZATION_ERROR a v protokolech chyb správce front se zobrazí zpráva AMQ9673 .

Další informace o tom, jak produkt IBM MQ poskytuje více funkcí certifikátu, naleznete v tématu Jak IBM MQ poskytuje více možností certifikátů .

Vývoj aplikací XMS .NET

IBM MQ Message Service Client (XMS) for .NET (XMS .NET) poskytuje rozhraní API s názvem XMS , které má stejnou sadu rozhraní jako Java Message Service (JMS) Rozhraní API. IBM MQ Message Service Client (XMS) for .NET obsahuje plně spravovanou implementaci XMS, kterou může používat libovolný .NET vyhovující jazyk.

Informace o této úloze

XMS podporuje:

- Dvoubodový systém zpráv
- Systém zpráv publikování/odběru
- Synchronní doručování zpráv
- Asynchronní doručování zpráv

Aplikace XMS si může vyměňovat zprávy s následujícími typy aplikací:

- Aplikace XMS
- Aplikace IBM MQ classes for JMS
- Nativní aplikace IBM MQ
- Aplikace JMS, která používá výchozího poskytovatele systému zpráv IBM MQ .

Aplikace XMS se může připojit a používat prostředky libovolného z následujících serverů systému zpráv:

IBM MQ správce front
Aplikace se může připojit buď v režimu vazeb, nebo v režimu klienta.

WebSphere Application Server service integration bus
Aplikace může používat přímé připojení TCP/IP nebo HTTP přes TCP/IP.

IBM Integration Bus
Zprávy jsou přenášeny mezi aplikací a zprostředkovatelem pomocí WebSphere MQ Real-Time Transport. Zprávy lze doručit do aplikace pomocí WebSphere MQ Multicast Transport.

Připojením ke správci front IBM MQ může aplikace XMS používat produkt WebSphere MQ Enterprise Transport ke komunikaci s produktem IBM Integration Bus. Alternativně může aplikace XMS publikovat a přihlásit se k odběru připojením k produktu IBM MQ.

Produkt IBM MQ 9.1.1 IBM MQ podporuje .NET Core pro aplikace v prostředí Windows . Další informace viz “Instalace produktu IBM MQ classes for XMS .NET” na stránce 602.

Produkt IBM MQ 9.1.2 IBM MQ podporuje .NET Core pro aplikace v prostředí Linux .

V produktu IBM MQ 9.1.4jsou spravované aplikace XMS .NET schopny automaticky vyvážit připojení mezi klastrovanými správci front. Podporovány jsou knihovny .NET Framework i .NET Standard . Další informace naleznete v tématu O uniformních klastrech a Automatické vyrovnávání aplikací.

Související úlohy
Kontaktování podpory společnosti IBM
Odstraňování problémů s produktem XMS .NETproblems

Styly systému zpráv podporované produktem XMS

Produkt XMS podporuje styly dvoubodového a publikování/odběru systému zpráv.

Styly systému zpráv se také nazývají domény systému zpráv.

Dvoubodový systém zpráv

Obvyklá forma systému zpráv typu point-to-point používá řazení do fronty. V nejjednodušším případě aplikace odešle zprávu jiné aplikaci tak, že implicitně nebo explicitně identifikuje cílovou frontu. Základní systém zpráv a systém front přijme zprávu od odesílající aplikace a směruje ji do cílové fronty. Přijímající aplikace pak může načíst zprávu z fronty.

Pokud základní systém zpráv a systém front obsahuje IBM Integration Bus, IBM Integration Bus může replikovat zprávu a směrovat její kopie do různých front. V důsledku toho může zprávu přijmout více než jedna aplikace. Produkt IBM Integration Bus může také transformovat zprávu a přidat do ní data.

Klíčovou charakteristikou systému zpráv typu point-to-point je, že aplikace při odesílání zprávy umístí zprávu do lokální fronty. Základní systém zpráv a systém front určuje, do které cílové fronty je zpráva odeslána. Přijímající aplikace načte zprávu z cílové fronty.

Systém zpráv publikování/odběru

V systému zpráv publikování/odběru existují dva typy aplikací: vydavatel a odběratel.

Vydavatel dodává informace ve formě zpráv publikování. Když vydavatel publikuje zprávu, uvádí téma, které identifikuje předmět informací uvnitř zprávy.

Odběratel je odběratel informací, které jsou publikovány. Odběratel určuje témata, o která má zájem, vytvořením odběrů.

Systém publikování/odběru přijímá publikace od vydavatelů a odběry od odběratelů. Směruje publikování k odběratelům. Odběratel přijímá publikování pouze v těch tématech, k jejichž odběru se přihlásil.

Klíčovou charakteristikou systému zpráv publikování/odběru je, že vydavatel identifikuje téma při publikování zprávy. Neidentifikuje odběratele. Pokud je zpráva publikována v tématu, pro které nejsou žádní odběratelé, žádná aplikace zprávu neobdrží.

Aplikace může být vydavatelem i odběratelem.

Objektový model XMS

Rozhraní XMS API je objektově orientované rozhraní. Model objektů XMS je založen na modelu objektů JMS 1.1 .

Hlavní třídy XMS

Hlavní třídy XMS nebo typy objektů jsou následující:

ConnectionFactory
Objekt `ConnectionFactory` zapouzdřuje sadu parametrů pro připojení. Aplikace používá `ConnectionFactory` k vytvoření připojení. Aplikace může poskytnout parametry za běhu a vytvořit objekt `ConnectionFactory` . Případně lze parametry připojení uložit do úložiště spravovaných objektů. Aplikace může načíst objekt z úložiště a vytvořit z něj objekt `ConnectionFactory` .

Připojení
Objekt `Connection` zapouzdřuje aktivní připojení z aplikace na server systému zpráv. Aplikace používá připojení k vytváření relací.

Místo určení
Aplikace odesílá nebo přijímá zprávy pomocí objektu `Destination` . V doméně publikování/odběru objekt `Destination` zapouzdřuje téma a v doméně typu point-to-point objekt `Destination` zapouzdřuje frontu. Aplikace může poskytnout parametry pro vytvoření objektu `Destination` za běhu. Případně může vytvořit objekt `Destination` z definice objektu, která je uložena v úložišti spravovaných objektů.

Relace
Objekt `Session` je kontext s jedním podprocesem pro odesílání a příjem zpráv. Aplikace používá objekt `Session` k vytvoření objektů `Message`, `MessageProducera` `MessageConsumer` .

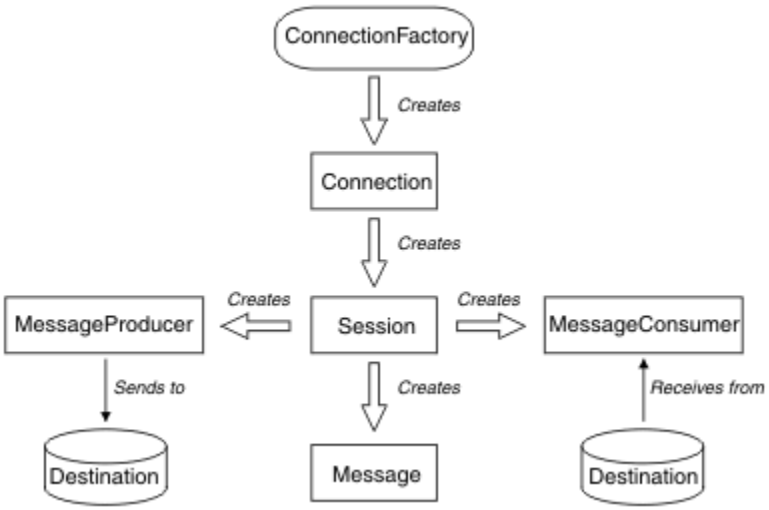
Zpráva
Objekt `Message` zapouzdřuje objekt `Message` , který aplikace odesílá pomocí objektu `MessageProducer` , nebo přijímá pomocí objektu `MessageConsumer` .

MessageProducer
Objekt `MessageProducer` je používán aplikací k odesílání zpráv do místa určení.

MessageConsumer
Objekt `MessageConsumer` je používán aplikací pro příjem zpráv odeslaných do místa určení.

Objekty XMS a jejich vztahy

Obrázek 52 na stránce 601 zobrazuje hlavní typy objektu XMS : ConnectionFactory, Connection, Session, MessageProducer, MessageConsumer, Message a Destination. Aplikace používá továrnu připojení k vytvoření připojení a používá připojení k vytvoření relací. Aplikace pak může pomocí relace vytvářet zprávy, producenty zpráv a spotřebitele zpráv. Aplikace používá producenta zpráv k odesílání zpráv do místa určení a spotřebitele zpráv používá k přijímání zpráv odeslaných do místa určení.



Obrázek 52. Objekty XMS a jejich vztahy

V produktu XMS .NETjsou třídy XMS definovány jako sada rozhraní .NET . Při kódování aplikací XMS .NET potřebujete pouze deklarovaná rozhraní.

Model objektů XMS je založen na rozhraních nezávislých na doméně, která jsou popsána v tématu Java Message Service Specifikace, verze 1.1. Nejsou poskytnuty třídy specifické pro doménu, jako např. `Topic`, `TopicPublishera` `TopicSubscriber`.

Atributy a vlastnosti objektů XMS

Objekt XMS může mít atributy a vlastnosti, což jsou charakteristiky objektu, které jsou implementovány různými způsoby:

Atributy
Charakteristika objektu, která je vždy přítomna a zabírá úložiště, i když atribut nemá hodnotu. V tomto ohledu je atribut podobný poli v datové struktuře s pevnou délkou. Charakteristickým rysem atributů je, že každý atribut má své vlastní metody pro nastavení a získání své hodnoty.

Vlastnosti
Vlastnost objektu je přítomna a zabírá úložiště pouze po nastavení jeho hodnoty. Vlastnost nelze odstranit nebo obnovit její úložiště po nastavení její hodnoty. Můžete změnit její hodnotu. Produkt XMS poskytuje sadu generických metod pro nastavení a získání hodnot vlastností.

Spravované objekty

Pomocí spravovaných objektů můžete spravovat nastavení připojení používaná klientskými aplikacemi pro správu z centrálního úložiště. Aplikace načte definice objektů z centrálního úložiště a použije je k vytvoření objektů `ConnectionFactory` a `Destination` . Pomocí spravovaných objektů můžete odstranit párové aplikace z prostředků, které používají za běhu.

Například aplikace XMS lze psát a testovat se spravovanými objekty, které odkazují na sadu připojení a cílů v testovacím prostředí. Při implementaci aplikací lze spravované objekty změnit tak, aby konfigurovaly aplikace tak, aby odkazovaly na připojení a cíle v produkčním prostředí.

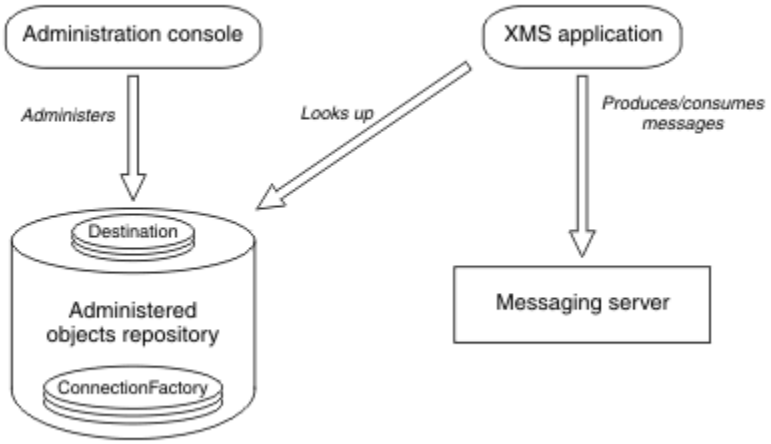
Produkt XMS podporuje dva typy spravovaných objektů:

- Objekt `ConnectionFactory` , který používají aplikace k vytvoření počátečního připojení k serveru.
- Objekt `Destination` používaný aplikacemi k určení místa určení pro odesílané zprávy a zdroje přijímaných zpráv. Místo určení je buď téma, nebo fronta na serveru, ke které se aplikace připojuje.

Nástroj pro administraci **`JMSAdmin`** je dodáván s produktem IBM MQ. Používá se k vytváření a správě spravovaných objektů v centrálním úložišti spravovaných objektů.

Spravované objekty v úložišti mohou být použity aplikacemi IBM MQ classes for JMS a XMS . Aplikace XMS mohou používat objekty `ConnectionFactory` a `Destination` pro připojení ke IBM MQ správci front. Administrátor může změnit definice objektů uložené v úložišti bez ovlivnění kódu aplikace.

Následující diagram ukazuje, jak aplikace XMS obvykle používá spravované objekty. Na levé straně diagramu je zobrazeno úložiště obsahující definice objektů ConnectionFactory a Destination, které jsou spravovány pomocí konzoly pro správu. Na pravé straně diagramu je zobrazena aplikace XMS , která vyhledává definice objektů v úložišti a poté tyto definice objektů používá při připojování k serveru systému zpráv.



Obrázek 53. Typické použití spravovaných objektů aplikací XMS

Model zpráv XMS

Model zpráv XMS je stejný jako model zpráv IBM MQ classes for JMS .

Produkt XMS konkrétně implementuje stejná pole záhlaví zprávy a vlastnosti zprávy, které produkt IBM MQ classes for JMS implementuje:

- Pole záhlaví JMS . Tato pole mají názvy, které začínají předponou JMS.
- JMS definované vlastnosti. Tato pole mají vlastnosti, jejichž názvy začínají předponou JMSX.
- IBM definované vlastnosti. Tato pole mají vlastnosti, jejichž názvy začínají předponou JMS_IBM_.

V důsledku toho si mohou aplikace XMS vyměňovat zprávy s aplikacemi IBM MQ classes for JMS . V každé zprávě jsou některá pole záhlaví a vlastnosti nastaveny aplikací a ostatní jsou nastaveny pomocí XMS nebo IBM MQ classes for JMS. Některá pole nastavená pomocí XMS nebo IBM MQ classes for JMS jsou nastavena při odeslání zprávy a jiná při jejím přijetí. Pole záhlaví a vlastnosti jsou šířeny se zprávou prostřednictvím serveru systému zpráv, kde je to vhodné. Jsou zpřístupněny pro všechny aplikace, které obdrží zprávu.

Související pojmy

IBM MQ classes for JMS

Windows Linux Instalace produktu IBM MQ classes for XMS .NET

Produkt IBM MQ classes for XMS .NET, včetně ukázek, se instaluje s produktem IBM MQ v systémech Windows a Linux.

Od verze IBM MQ 9.2.0je Microsoft.NET Core 3.1 minimální požadovaná verze pro spuštění IBM MQ classes for XMS .NET Standard.

V 9.3.0 V 9.3.0 V systému IBM MQ 9.3.0 IBM MQ podporuje .NET 6 aplikace používající IBM MQ classes for XMS .NET Standard. Používáte-li aplikaci .NET Core 3.1 , můžete spustit tuto aplikaci

s malou úpravou v souboru `csproj` , nastavením `targetframeworkversion` na hodnotu `"net6.0"`, bez nutnosti rekompilace.

V 9.3.1 Produkt IBM MQ 9.3.1 poskytuje knihovnu klienta XMS .NET sestavenou pro .NET 6 jako cílový rámec. V produktu IBM MQ 9.3.1je Microsoft .NET 6.0 minimální požadovaná verze pro spouštění aplikací pomocí knihoven IBM MQ , které jsou sestaveny pomocí .NET 6 jako cílový rámec.

V 9.3.1 V systému IBM MQ 9.3.1je knihovna klienta XMS .NET sestavená pomocí .NET Standard k dispozici v nové složce `netstandard2.0` a knihovna klienta XMS .NET sestavená pomocí .NET 6 jako cílový rámec je k dispozici v systému *MQ_INSTALLATION_PATH*`/bin` v systému Windows a v systému *MQ_INSTALLATION_PATH*`/lib64` v systému Linux.

`amqmxmsstd.dll` knihovna

V 9.3.1 V produktu IBM MQ 9.3.1je knihovna `amqmxmsstd.dll` k dispozici v následujících umístěních.

Knihovna sestavená pomocí .NET Standard 2.0 jako cílového rámce

- Windows V systému Windows: *MQ_INSTALLATION_PATH*`\bin\netstandard2.0`.
- Linux V systému Linux: *MQ_INSTALLATION_PATH*`\lib64\netstandard2.0`.

Deprecated Tyto knihovny jsou zamítnuté a produkt IBM je hodlá v budoucích verzích odebrat.

Knihovna sestavená pomocí .NET 6 jako cílového rámce

- Windows V systému Windows: *MQ_INSTALLATION_PATH*`\bin`. Ukázkové aplikace jsou nainstalovány v produktu *MQ_INSTALLATION_PATH*`/samp/dotnet/samples/cs/core/base`.
- Linux V systému Linux: *MQ_INSTALLATION_PATH*`\lib64`. Ukázky .NET jsou v adresáři *MQ_INSTALLATION_PATH*`/samp/dotnet/samples/cs/core/base`.

LTS V případě operačního systému IBM MQ 9.3.0 Long Term Supportje knihovna IBM MQ classes for XMS .NET Standard `amqmxmsstd.dll`k dispozici v následujících umístěních:

- Windows V systému Windows: *MQ_INSTALLATION_PATH*`\bin`. Ukázkové aplikace jsou nainstalovány v produktu *MQ_INSTALLATION_PATH*`/samp/dotnet/samples/cs/core/xms`.
- Linux V systému Linux: *MQ_INSTALLATION_PATH*`/lib64 path`. Ukázky .NET jsou v adresáři *MQ_INSTALLATION_PATH*`/samp/dotnet/samples/cs/core/xms`.

Další informace viz téma “Instalace produktu IBM MQ classes for .NET” na stránce 541.

Upozornění: Deprecated V 9.3.1 Z IBM MQ 9.3.1, IBM MQ .NET klientské knihovny sestavené pomocí .NET Standard 2.0 jako cílový rámec jsou zamítnuty a aplikace odkazující na tyto knihovny vygenerují během doby kompilace varování CS0618 .

Stabilized Všechny knihovny `IBM.XMS.*` jsou stále dodávány, ale tyto knihovny jsou stabilizované; to znamená, že do nich nebudou zavedeny žádné nové funkce. Pro všechny nejnovější funkce musíte provést migraci do knihovny `amqmxmsstd.dll` . Můžete však i nadále používat existující knihovny ve vydáních IBM MQ 9.1 Long Term Support nebo Continuous Delivery .

V 9.3.1 Pokud je aplikace .NET Framework kompilována pomocí `amqmdnetstd.dll` nebo `amqmxmsstd.dll` z verze nižší než IBM MQ 9.3.1 a stejná aplikace je spuštěna pomocí knihoven klienta .NET 6 založených na IBM MQ , pak .NETvygeneruje výjimku typu `FileLoadException` :

```
Byla zachycena výjimka: System.IO.FileLoadException: Nelze načíst soubor nebo sestavení
'amqmdnetstd, Version =x.x.x.x, Culture=neutral, PublicKeyToken=23d6cb914eeaac0e' nebo
jedna z jeho závislostí. Definice souboru typu manifest umístěného sestavení neodpovídá
odkaz na sestavení. (Výjimka z HRESULT: 0x80131040)
```

```
Název souboru: ' amqmdnetstd, Version =x.x.x.x, Culture=neutral,
PublicKeyToken=23d6cb914eeaac0e'
```

Chcete-li vyřešit tuto chybu, knihovny přítomné v produktu *MQ_INSTALLATION_PATH*`/bin/netstandard2.0` musí být zkopírovány do adresáře, ze kterého je spuštěna aplikace .NET Framework .

V produktu IBM MQ 9.2.0je soubor IBM MQ classes for XMS .NET Standard k dispozici pro stažení z úložiště NuGet . Balík NuGet obsahuje knihovnu `amqmxmsstd.dll` i knihovnu `amqmdnetstd.dll` . Produkt `amqmxmsstd.dll` je závislý na produktu `amqmdnetstd.dll` a při balení aplikace XMS .NET Core by měly být zabaleny jak `amqmxmsstd.dll` , tak `amqmdnetstd.dll` spolu s aplikací XMS .NET Core. Další informace viz “Stažení souboru IBM MQ classes for XMS .NET z úložiště NuGet” na stránce 605.

`dspmqver` příkaz

Pomocí příkazu **`dspmqver`** můžete zobrazit informace o verzi a sestavení pro komponentu .NET Core .

Porovnání mezi knihovnami IBM MQ classes for XMS .NET Framework a IBM MQ classes for XMS .NET (.NET Standard a .NET 6)

V následující tabulce jsou uvedeny funkce pro produkt IBM MQ classes for XMS .NET Framework v porovnání s funkcemi pro knihovny IBM MQ classes for XMS .NET (.NET Standard a .NET 6).

Tabulka 80. Rozdíly mezi knihovnami IBM MQ classes for XMS .NET Framework a IBM MQ classes for XMS .NET (.NET Standard a .NET 6)

Funkce	**IBM MQ classes for XMS .NET Framework**	**IBM MQ classes for XMS .NET (knihovny.NET Standard a .NET 6)**
Názvy tříd (API)	Všechny třídy zůstávají stejné v každé síti.	Všechny třídy zůstávají stejné v každé síti.
Operační systém	Windows	Windows Kontejnery v nástroji Docker Linux macOS
soubor `app.config` (konfigurační soubor pro povolení trasování v redistribuovatelném klientovi)	Soubor `app.config` se používá k povolení trasování pro redistribuovatelný balík.	`app.config` není podporován. Použijte proměnné prostředí.
Trasovat	Chcete-li trasovat klienta XMS .NET , můžete použít existující proměnné prostředí, jako např. proměnnou prostředí **`XMS_TRACE_ON`** , která se používá k povolení trasování. Další informace naleznete v tématu Konfigurace trasování produktu XMS .NET pomocí XMS proměnných prostředí. Pro redistribuovatelné klienty lze k povolení trasování použít soubor `app.config` .	Chcete-li trasovat klienta XMS .NET , můžete použít existující proměnné prostředí, jako např. proměnnou prostředí **`XMS_TRACE_ON`** , která se používá k povolení trasování. Další informace naleznete v tématu Konfigurace trasování produktu XMS .NET pomocí XMS proměnných prostředí.
Přenosové režimy	Spravované, nespravované a vazby	Spravováno

Tabulka 80. Rozdíly mezi knihovnami IBM MQ classes for XMS .NET Framework a IBM MQ classes for XMS .NET (.NET Standard a .NET 6) (pokračování)

Funkce	IBM MQ classes for XMS .NET Framework	IBM MQ classes for XMS .NET (knihovny.NET Standard a .NET 6)
TLS	Úložiště klíčů Windows se používá k ukládání certifikátů.	Úložiště klíčů Windows musí být použito k uložení certifikátů. Povolené hodnoty jsou *USER nebo *SYSTEM. Na základě vstupu se klient produktu IBM MQ .NET podívá do úložiště klíčů Windows aktuálního uživatele, nebo se podívá do celého systému. Na systému Linux se k instalaci certifikátů doporučuje použít třídu X509Store a produkt .NET Core nainstaluje certifikáty do následujícího umístění: "`.dotnet/corefx/cryptography/x509stores`".
CCDT	Podporováno	Podporováno a nastavení cesty CCDT jsou stejná jako pro třídy .NET Framework.
Automatické znovupřipojení klienta	Podporováno	Podporováno
Distribuované transakce	Podporováno	Nepodporováno
Instalace dynamicky propojených knihoven (dll) do globální mezipaměti sestavení (GAC)	Knihovny Dll jsou nainstalovány do GAC jako součást instance IBM MQ.	Knihovny Dll nejsou nainstalovány do GAC jako součást instance IBM MQ.
Podpora typů připojení WMQ, WPM a RTT	Podporuje typy připojení WMQ, WPM a RTT.	Podpora pouze pro WMQ
Spravované objekty rozhraní JNDI	Podporuje služby LDAP a FileSystem .	Podporuje pouze volbu FileSystem .

V 9.3.0 V 9.3.0 Chcete-li spustit produkt IBM MQ classes for XMS .NET Framework z adresáře IBM MQ 9.3.0, musíte nainstalovat produkt Microsoft.NET Framework V4.7.2 nebo novější.

Související úlohy

“Použití ukázkových aplikací XMS” na stránce 612
Ukázkové aplikace produktu XMS .NET poskytují přehled společných funkcí jednotlivých rozhraní API. Můžete je použít k ověření instalace a nastavení serveru systému zpráv a k usnadnění vytváření vlastních aplikací.

Windows Linux Stažení souboru IBM MQ classes for XMS .NET z úložiště NuGet

Produkt IBM MQ classes for XMS .NET je k dispozici pro stažení z úložiště NuGet , aby jej mohli snadno využívat vývojáři produktu .NET .

Informace o této úloze

NuGet je správce balíků pro Microsoft vývojové platformy včetně .NET. Nástroje klienta NuGet poskytují schopnost vytvářet a spotřebovávat balíky. Balík NuGet je jediný komprimovaný soubor s příponou `.nupkg` , který obsahuje kompilovaný kód (DLL), další soubory související s tímto kódem a popisný soubor typu manifest, který obsahuje informace, jako je číslo verze balíku.

Balík `IBMXMSDotnetClient` NuGet , který obsahuje knihovnu `amqmdnetstd.dll` i knihovnu `amqmxmsstd.dll` , si můžete stáhnout z galerie NuGet , což je centrální úložiště balíků používané všemi autory a spotřebiteli balíků.

Poznámka: V 9.3.1 V produktu IBM MQ 9.3.1obsahuje balík NuGet knihovny sestavené pomocí .NET Standard 2.0 a .NET 6 jako cílový rámec. Knihovny pro produkt .NET Standard 2.0 jsou k dispozici ve složce `netstandard2.0` a knihovny pro produkt .NET 6 jsou k dispozici ve složce `net6.0` .

Existují tři způsoby stažení balíku `IBMXMSDotnetClient` :

- Pomocí Microsoft Visual Studio. Produkt NuGet je distribuován jako rozšíření Microsoft Visual Studio . V produktu Microsoft Visual Studio 2012je produkt NuGet standardně předinstalován.
- Z příkazového řádku buď pomocí produktu NuGet Package Manager, nebo pomocí rozhraní příkazového řádku .NET .
- Pomocí webového prohlížeče.

Pokud jde o redistribuovatelný balík, povolíte trasování pomocí proměnné prostředí **XMS_TRACE_ON**.

Procedura

- Chcete-li stáhnout balík `IBMXMSDotnetClient` pomocí uživatelského rozhraní správce balíků v produktu Microsoft Visual Studio, postupujte takto:
  a) Klepněte pravým tlačítkem myši na projekt .NET a poté klepněte na volbu **Spravovat balíky Nuget**.
  b) Klepněte na kartu **Procházet** a vyhledejte `"IBMXMSDotnetClient"`.
  c) Vyberte balík a klepněte na tlačítko **Instalovat**.
  Během instalace poskytuje správce balíků informace o průběhu ve formě příkazů konzoly.
- Chcete-li stáhnout balík `IBMXMSDotnetClient` z příkazového řádku, vyberte jednu z následujících voleb:
  - Pomocí správce balíků NuGet zadejte následující příkaz:

    ```
    Install-Package IBMXMSDotnetClient -Version 9.1.4.0
    ```

    Během instalace poskytuje správce balíků informace o průběhu ve formě příkazů konzoly. Výstup můžete přesměrovat do souboru protokolu.
  - Pomocí rozhraní příkazového řádku .NET zadejte následující příkaz:

    ```
    dotnet add package IBMXMSDotnetClient --version 9.1.4
    ```
- Pomocí webového prohlížeče stáhněte balík `IBMXMSDotnetClient` z adresáře https://www.nuget.org/packages/IBMXMSDotnetClient.

Související pojmy

“Instalace produktu IBM MQ classes for .NET” na stránce 541
Produkt IBM MQ classes for .NET, včetně ukázek, se instaluje s operačním systému IBM MQ na systémech Windows a Linux .

IBM MQ Klient pro .NET informace o licenci

Související úlohy

“Stažení IBM MQ classes for .NET z úložiště NuGet” na stránce 546
IBM MQ classes for .NET je k dispozici ke stažení z úložiště NuGet , aby je mohli snadno využívat vývojáři produktu .NET .

Nastavení prostředí serveru systému zpráv

Témata v této části popisují, jak nastavit prostředí serveru systému zpráv tak, aby umožňovalo aplikacím XMS připojit se k serveru.

Informace o této úloze

Pro aplikace, které se připojují ke správci front IBM MQ , je vyžadován klient IBM MQ (nebo správce front pro režim vazeb).

V současné době neexistují žádné nezbytné předpoklady pro aplikace, které používají připojení v reálném čase ke zprostředkovateli.

Před spuštěním jakýchkoli aplikací XMS , včetně ukázkových aplikací dodávaných s produktem XMS, musíte nastavit prostředí serveru systému zpráv.

Tento oddíl obsahuje následující témata:

- "Konfigurace správce front a zprostředkovatele pro aplikaci, která se připojuje ke správci front IBM MQ" na stránce 609
- "Instalace produktu IBM MQ classes for XMS .NET" na stránce 602
- "Konfigurace zprostředkovatele pro aplikaci, která používá připojení v reálném čase ke zprostředkovateli" na stránce 610
- "Konfigurace sběrnice SIBus pro aplikaci, která se připojuje k produktu WebSphere Application Server" na stránce 611

Moduly listener pro zprávy v adresáři XMS .NET

Modul listener pro zprávy se používá k asynchronnímu příjmu zpráv. Na rozdíl od volání `MessageConsumer.receive()` modul listener pro zprávy neblokuje volající podproces, místo toho doručí zprávy do metody zpětného volání určené aplikací, obvykle do metody `onMessage` .

Doručení zprávy se spustí po volání metody `Connection.Start()` . Doručování zpráv lze kdykoli zastavit a obnovit pomocí metod `Connection.Stop()` a `Connection.Start()` .

Po volání metody `Connection.Start()` po nastavení modulu listener pro zprávy na alespoň jednoho spotřebitele v relaci se tato relace stane asynchronní relací. Jakmile se relace stane asynchronní, není možné volat žádné XMS .NET synchronní metody., Například `MessageProducer.Send()`. Výsledkem je výjimka s kódem příčiny IBM MQ MQRC_HCONN_ASYNC_ACTIVE (2500).

Synchronní volání v asynchronní relaci

`Session.Close` je jediné synchronní volání, které je povoleno v asynchronní relaci. Aplikace mohou také provádět synchronní volání (s výjimkou volání `Session.Close`) pomocí metody zpětného volání modulu listener pro zprávy, tj. metody `onMessage` .

Kromě těchto dvou voleb musíte zastavit připojení pomocí metody `Connection.Stop()` pro aplikaci, aby bylo možné provést jakékoli synchronní volání. Po provedení volání musíte znovu obnovit připojení pomocí metody `Connection.Start()` . který restartuje doručování zpráv.

Kolik asynchronních spotřebitelů zpráv může relace mít?

Relace může mít více asynchronních spotřebitelů zpráv. Kdykoli je však zpráva doručena pouze jednomu spotřebiteli. To prakticky znamená, že když dorazí druhá zpráva, zatímco produkt XMS .NET zavolá metodu `onMessage()` spotřebitele, aby doručil první zprávu, druhá zpráva nebude doručena spotřebiteli v relaci, dokud se nevrátí metoda `onMessage()` .

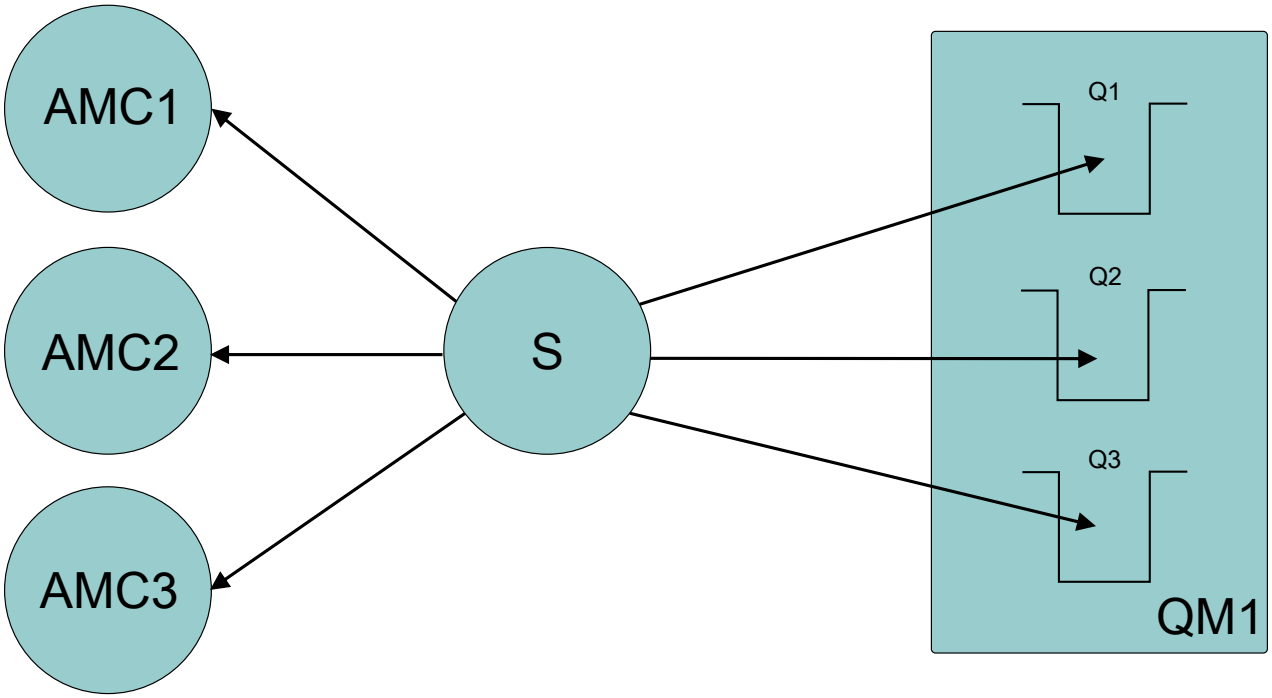
Druhá zpráva je doručena spotřebiteli v relaci pouze po vrácení metody `onMessage()` . Důvodem je, že relace spravuje doručování zpráv spotřebitelům pouze pomocí jednoho podprocesu. To znamená, že v daném okamžiku může být doručena pouze jedna zpráva a spotřebitel může být libovolná.

Pokud aplikace vyžaduje souběžné doručení zpráv, tj. všichni spotřebitelé musí přijímat zprávy současně, musí aplikace vytvořit více relací a každý z nich musí mít jednoho asynchronního spotřebitele zpráv.

Následující příklady ukazují tuto funkci jasněji.

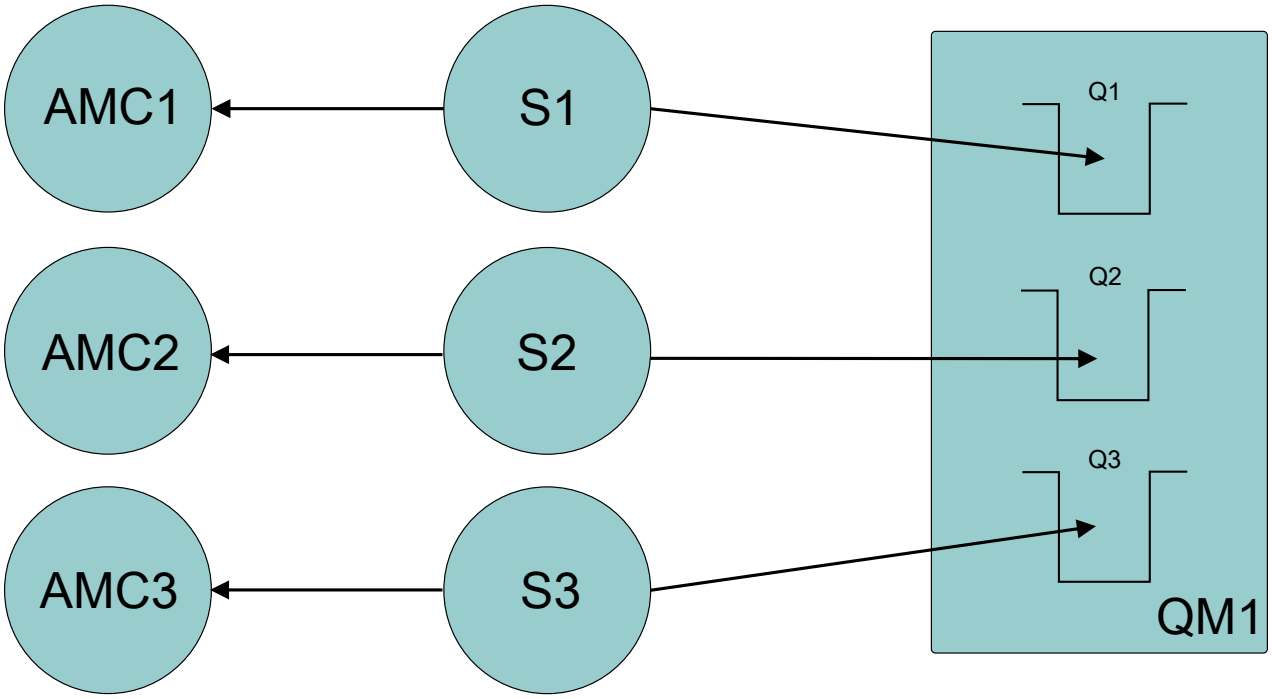
V prvním příkladu je v relaci více asynchronních spotřebitelů zpráv. Relace S má tři asynchronní spotřebitele zpráv: AMC1, AMC2 a AMC3 , kteří přijímají zprávy ze tří různých míst určení Q1, Q2 a Q3.

Protože existuje pouze jedna relace S, existuje pouze podproces doručení zpráv, který doručuje zprávy spotřebitelům AMC1, AMC2a AMC3. Když relace doručuje zprávu do produktu AMC1, ostatní dva spotřebitelé AMC2 a AMC3 čekají, i když jsou zprávy v adresáři Q2 a Q3 připraveny k doručení.



Obrázek 54. Jedna relace se třemi asynchronními spotřebiteli zpráv

Ve druhém případě existuje více relací S1, S2a S3, z nichž každá má jednoho asynchronního spotřebitele zpráv AMC1, AMC2a AMC3 . Vzhledem k tomu, že pro každou relaci existuje jeden spotřebitel, jsou zprávy doručovány spotřebitelům souběžně.



Obrázek 55. Více relací, každá s jedním asynchronním spotřebitelem zpráv

To ukazuje, že pokud požadujete souběžné doručení zpráv, potřebujete více relací.

Konfigurace správce front a zprostředkovatele pro aplikaci, která se připojuje ke správci front IBM MQ

Tato část předpokládá, že používáte produkt IBM WebSphere MQ 7.0.1nebo novější. Před spuštěním aplikace, která se připojuje ke správci front IBM MQ , je třeba konfigurovat správce front. V případě aplikace publikování/odběru je vyžadována nějaká další konfigurace, pokud používáte rozhraní pro publikování/odběr ve frontě.

Než začnete

XMS pracuje s produktem IBM Integration Bus nebo WebSphere Message Broker 6.1 nebo novějším

Před spuštěním této úlohy proveďte následující kroky:

- Ujistěte se, že vaše aplikace má přístup ke spuštěnému správci front.
- Pokud je vaše aplikace aplikací typu publikování/odběr a používá rozhraní pro publikování/odběr ve frontě, zkontrolujte, zda je atribut **PSMODE** ve správci front nastaven na hodnotu POVOLENO .
- Přesvědčte se, že vaše aplikace používá továrnu připojení, jejíž vlastnosti jsou správně nastaveny pro připojení ke správci front. Pokud je vaše aplikace aplikací typu publikování/odběr, ujistěte se, že jsou pro použití zprostředkovatele nastaveny příslušné vlastnosti továrny připojení. Další informace o vlastnostech továrny připojení naleznete v tématu Vlastnosti továrny připojení ConnectionFactory.

Informace o této úloze

Správce front a zprostředkovatele konfigurujete tak, aby spouštěli aplikace XMS stejným způsobem, jakým konfigurujete správce front a rozhraní pro publikování/odběr ve frontě, aby spouštěli aplikace IBM MQ JMS . Následující kroky shrnují, co je třeba udělat.

Postup

1. Ve správci front vytvořte fronty, které vaše aplikace potřebuje.

   Chcete-li získat přehled o tom, jak vytváříte fronty, prohlédněte si téma Definování front.

Pokud je vaše aplikace aplikací typu publikování/odběr a používá rozhraní pro publikování/odběr ve frontě, které potřebuje přístup k systémovým frontám IBM MQ classes for JMS , počkejte před vytvořením front na krok 4a .

2. Udělte uživateli přidruženému k aplikaci oprávnění pro připojení ke správci front a příslušné oprávnění pro přístup k frontám.

   Přehled o autorizaci naleznete v tématu Zabezpečení. Pokud se vaše aplikace připojuje ke správci front v režimu klienta, viz také téma Klienti a servery.

3. Pokud se vaše aplikace připojuje ke správci front v režimu klienta, zkontrolujte, zda je ve správci front definován kanál připojení serveru a zda je spuštěn modul listener.

   Tento krok není nutné provádět pro každou aplikaci, která se připojuje ke správci front. Jedna definice kanálu připojení serveru a jeden modul listener mohou podporovat všechny aplikace, které se připojují v režimu klienta.

4. Pokud je vaše aplikace aplikací typu publikování/odběr a používá rozhraní pro publikování/odběr ve frontě, postupujte takto.

   a) Ve správci front vytvořte systémové fronty IBM MQ classes for JMS spuštěním skriptu příkazů MQSC dodávaných s produktem IBM MQ. Ujistěte se, že ID uživatele přidružené k IBM Integration Bus nebo WebSphere Message Broker má oprávnění pro přístup k frontám.

      Informace o tom, kde skript vyhledat a jak jej spustit, viz Použití IBM MQ classes for Java.

      Tento krok proveďte pouze jednou pro správce front. Stejná sada systémových front IBM MQ classes for JMS může podporovat všechny aplikace XMS a IBM MQ classes for JMS , které se připojují ke správci front.

   b) Udělte ID uživatele přidruženému k aplikaci oprávnění pro přístup k systémovým frontám IBM MQ classes for JMS .

      Chcete-li získat informace o tom, jaká oprávnění ID uživatele potřebuje, prohlédněte si téma Použití IBM MQ classes for JMS.

   c) Pro zprostředkovatele IBM Integration Bus nebo WebSphere Message Brokervytvořte a implementujte tok zpráv do fronty, kam aplikace odesílají zprávy, které publikují.

      Základní tok zpráv se skládá z uzlu zpracování zpráv MQInput pro čtení publikovaných zpráv a z uzlu zpracování publikovaných zpráv pro publikování zpráv.

      Chcete-li získat informace o tom, jak vytvořit a implementovat tok zpráv, prohlédněte si dokumentaci k produktu IBM Integration Bus nebo WebSphere Message Broker , která je k dispozici v produktu IBM Integration Bus webová stránka knihovny dokumentace produktu.

      Tento krok není třeba provádět, pokud je ve zprostředkovateli již implementován vhodný tok zpráv.

Výsledky

Nyní můžete spustit svou aplikaci.

Konfigurace zprostředkovatele pro aplikaci, která používá připojení v reálném čase ke zprostředkovateli

Než budete moci spustit aplikaci, která používá připojení v reálném čase ke zprostředkovateli, musíte tohoto zprostředkovatele nakonfigurovat.

Než začnete

Před spuštěním této úlohy proveďte následující kroky:

- Ujistěte se, že vaše aplikace má přístup ke zprostředkovateli, který je spuštěn.
- Ujistěte se, že vaše aplikace používá továrnu připojení, jejíž vlastnosti jsou správně nastaveny pro připojení v reálném čase ke zprostředkovateli. Další informace o vlastnostech továrny připojení naleznete v tématu Vlastnosti továrny připojení ConnectionFactory.

Informace o této úloze

Zprostředkovatele nakonfigurujete tak, aby spouštěli aplikace XMS stejným způsobem, jakým konfigurujete zprostředkovatele pro spouštění aplikací IBM MQ classes for JMS . Následující kroky shrnují, co je třeba udělat:

Postup

1. Vytvořte a implementujte tok zpráv pro čtení zpráv z portu TCP/IP, na kterém zprostředkovatel naslouchá, a publikujte zprávy.

   To můžete provést jedním z následujících způsobů:

   - Vytvořte tok zpráv, který obsahuje uzel zpracování zpráv **Real-timeOptimizedFlow** .
   - Vytvořte tok zpráv, který obsahuje uzel zpracování zpráv **Real-timeInput** a uzel zpracování zpráv publikování.

   Musíte nakonfigurovat uzel **Real-timeOptimizedFlow** nebo **Real-timeInput** , aby naslouchal na portu, který se používá pro připojení v reálném čase. V produktu XMSje výchozí číslo portu pro připojení v reálném čase 1506.

   Tento krok není třeba provádět, pokud je ve zprostředkovateli již implementován vhodný tok zpráv.
2. Pokud požadujete doručení zpráv do aplikace pomocí produktu IBM MQ classes for JMS, konfigurujte zprostředkovatele tak, aby umožňoval výběrové vysílání. Konfigurujte témata, pro která musí být povoleno výběrové vysílání, a určete spolehlivou kvalitu služeb pro témata vyžadující spolehlivé výběrové vysílání.
3. Pokud vaše aplikace při připojení ke zprostředkovateli zadá ID uživatele a heslo a vy chcete, aby zprostředkovatel ověřil vaši aplikaci pomocí těchto informací, nakonfigurujte server jmen uživatelů a zprostředkovatele pro jednoduché ověření pomocí hesla telnet.

Výsledky

Nyní můžete spustit svou aplikaci.

Konfigurace sběrnice SIBus pro aplikaci, která se připojuje k produktu WebSphere Application Server

Než budete moci spustit aplikaci, která se připojí ke sběrnici pro integraci služeb WebSphere Application Server service integration technologies , musíte nakonfigurovat integraci služeb stejným způsobem, jakým nakonfigurujete sběrnici pro integraci služeb ke spuštění aplikací JMS , které používají výchozího poskytovatele systému zpráv.

Než začnete

Před spuštěním této úlohy musíte provést následující kroky:

- Ujistěte se, že je vytvořena sběrnice systému zpráv a že je váš server přidán do sběrnice jako člen sběrnice.
- Ujistěte se, že vaše aplikace má přístup ke sběrnici pro integraci služeb, která obsahuje alespoň jeden spuštěný stroj systému zpráv.
- Je-li vyžadována operace HTTP , musí být definován příchozí transportní kanál stroje systému zpráv HTTP . Kanály pro SSL a TCP jsou standardně definovány během instalace serveru.
- Ujistěte se, že vaše aplikace používá továrnu připojení, jejíž vlastnosti jsou správně nastaveny pro připojení ke sběrnici SIBus pomocí serveru samozavedení. Minimální požadované informace jsou:
  - Koncový bod poskytovatele, který popisuje umístění a protokol, který se má použít při vyjednávání připojení k serveru systému zpráv (tj. prostřednictvím serveru samozavedení). Ve své nejjednodušší formě lze pro server nainstalovaný s výchozím nastavením nastavit koncový bod poskytnutí na název hostitele serveru.
  - Název sběrnice, přes kterou jsou odesílány zprávy.

Další informace o vlastnostech továrny připojení naleznete v tématu Vlastnosti továrny připojení ConnectionFactory.

Informace o této úloze

Všechny požadované prostory front nebo témat musí být definovány. Standardně je prostor tématu s názvem Default.Topic.Space definován během instalace serveru, ale pokud potřebujete další prostory tématu, musíte tyto prostory tématu vytvořit sami. Není třeba předdefinovat jednotlivá témata v rámci prostoru tématu, protože server vytváří instance těchto jednotlivých témat dynamicky podle potřeby.

Následující kroky shrnují, co je třeba udělat.

Postup

1. Vytvořte fronty, které vaše aplikace potřebuje pro systém zpráv typu point-to-point.
2. Vytvořte další prostory témat, které vaše aplikace potřebuje pro systém zpráv publikování/odběru.

Výsledky

Nyní můžete spustit svou aplikaci.

Použití ukázkových aplikací XMS

Ukázkové aplikace produktu XMS .NET poskytují přehled společných funkcí jednotlivých rozhraní API. Můžete je použít k ověření instalace a nastavení serveru systému zpráv a k usnadnění vytváření vlastních aplikací.

Informace o této úloze

Potřebujete-li pomoc při vytváření vlastních aplikací, můžete ukázkové aplikace použít jako výchozí bod. Pro každou aplikaci je k dispozici jak zdrojová, tak i zkompilovaná verze. Zkontrolujte ukázkový zdrojový kód a identifikujte klíčové kroky pro vytvoření každého požadovaného objektu pro vaši aplikaci (ConnectionFactory, Connection, Session, Destination a Producent nebo Konzument nebo obojí) a nastavte všechny specifické vlastnosti, které jsou zapotřebí k určení způsobu, jakým má vaše aplikace fungovat. Další informace viz téma “Psaní aplikací XMS” na stránce 615. Ukázky se mohou v budoucích verzích produktu XMSměnit.

Následující tabulka zobrazuje sady ukázkových aplikací (jednu pro každé rozhraní API), které jsou dodávány s produktem XMS.

Tabulka 81. Ukázkové aplikace pro produkt XMS .NET

Název vzorku	**Popis**
SampleConsumerCS	Aplikace spotřebitele zpráv, která přijímá zprávy z fronty nebo odebírá téma.
SampleProducerCS	Aplikace producenta zpráv, která vytváří zprávy do fronty nebo na téma.
SampleConfigCS	Konfigurační aplikace, kterou můžete použít k vytvoření úložiště spravovaných objektů, které je založeno na souborech. Aplikace obsahuje továrnu připojení a cíl pro vaše konkrétní nastavení připojení. Toto spravované úložiště objektů lze poté použít s každou ukázkovou aplikací odběratele a producenta.

Ukázky, které podporují stejné funkce v různých rozhraních API, mají syntaktické rozdíly.

- Ukázkový spotřebitel zpráv a aplikace producenta podporují následující funkce:
  - Připojení k IBM MQ, IBM Integration Bus (pomocí připojení v reálném čase ke zprostředkovateli) a WebSphere Application Server service integration bus

  - Vyhledání spravovaného úložiště objektů pomocí rozhraní počátečního kontextu
  - Připojení k frontám (IBM MQ a WebSphere Application Server service integration bus) a tématům (IBM MQ, připojení ke zprostředkovateli v reálném čase a WebSphere Application Server service integration bus)
  - Základní, bajtové, mapové, objektové, proudové a textové zprávy
- Ukázková aplikace spotřebitele zpráv podporuje synchronní a asynchronní režimy příjmu a příkazy selektoru SQL.
- Ukázková aplikace producenta zpráv podporuje trvalé a dočasné režimy doručení.

Vzorky mohou pracovat v jednom ze dvou režimů:

Jednoduchý režim
Ukázky můžete spustit s minimálním uživatelským vstupem.

rozšířený režim
Můžete přizpůsobit jemněji způsob, jakým ukázky fungují.

Všechny ukázky jsou kompatibilní, a proto mohou pracovat v různých jazycích.

Windows V produktu IBM MQ 9.1.1produkt IBM MQ podporuje .NET Core for XMS .NET aplikace v prostředích Windows . Produkt IBM MQ classes for .NET Standard, včetně ukázek, se standardně instaluje jako součást standardní instalace produktu IBM MQ .

Linux V produktu IBM MQ 9.1.2produkt IBM MQ také podporuje jádro .NET pro aplikace v prostředí Linux .

Ukázkové aplikace pro produkt XMS .NET jsou nainstalovány v adresáři `&MQINSTALL_PATH&/samp/dotnet/samples/cs/core/xms`.

Další informace viz téma “Instalace produktu IBM MQ classes for XMS .NET” na stránce 602.

Spuštění ukázkových aplikací .NET

Ukázkové aplikace .NET můžete spustit interaktivně buď v jednoduchém, nebo rozšířeném režimu, nebo neinteraktivně pomocí automaticky generovaných nebo upravených souborů odpovědí.

Než začnete

Před spuštěním libovolné z dodaných ukázkových aplikací musíte nejprve nastavit prostředí serveru systému zpráv tak, aby se aplikace mohly připojit k serveru. Viz “Nastavení prostředí serveru systému zpráv” na stránce 607.

Postup

Chcete-li spustit ukázkovou aplikaci .NET, postupujte takto:

Tip: Když spouštíte ukázkovou aplikaci, zadejte? kdykoli získat pomoc o tom, co dělat dál.

1. Vyberte režim, ve kterém chcete spustit ukázkovou aplikaci.
   Zadejte buď `Advanced` , nebo `Simple`.
2. Odpovězte na otázky.
   Chcete-li vybrat výchozí hodnotu, která je zobrazena v hranatých závorkách na konci otázky, stiskněte klávesu Enter. Chcete-li vybrat jinou hodnotu, zadejte odpovídající hodnotu a stiskněte klávesu Enter.
   Zde je příklad otázky:

   ```
   Enter connection type [wpm]:
   ```

   V tomto případě je výchozí hodnota `wpm` (připojení k serveru WebSphere Application Server service integration bus).

Výsledky

Když spustíte ukázkové aplikace, soubory odpovědí se automaticky vygenerují v aktuálním pracovním adresáři. Názvy souborů odpovědí jsou ve formátu *connection_type-sample_type*.rsp, například wpm-producer.rsp. V případě potřeby můžete vygenerovaný soubor odpovědí použít k opětovnému spuštění ukázkové aplikace se stejnými volbami, abyste tyto volby nemuseli zadávat znovu.

Související úlohy

Sestavení ukázkových aplikací .NET
Když sestavíte ukázkovou aplikaci .NET , vytvoří se spustitelná verze vybrané ukázky.

Sestavení vlastních aplikací
Sestavujete vlastní aplikace, jako byste sestavili ukázkové aplikace.

Sestavení ukázkových aplikací .NET

Když sestavíte ukázkovou aplikaci .NET , vytvoří se spustitelná verze vybrané ukázky.

Než začnete

Nainstalujte příslušný kompilátor. Tato úloha předpokládá, že jste nainstalovali produkt Microsoft Visual Studio 2012 a že jste s jeho použitím obeznámeni.

Postup

Chcete-li sestavit ukázkovou aplikaci .NET , postupujte takto:

1. Klepněte na soubor řešení Samples.sln poskytnutý s ukázkami .NET .
2. Klepněte pravým tlačítkem myši na řešení Ukázky v okně Průzkumník řešení a vyberte volbu **Sestavit řešení**.

Výsledky

Spustitelný program se vytvoří v příslušné podsložce ukázky, buď bin/Debug , nebo bin/Release, v závislosti na konfiguraci, kterou jste vybrali. Tento program má stejný název jako složka s příponou CS. Pokud například sestavujete verzi C# ukázkové aplikace producenta zpráv, vytvoří se SampleProducerCS.exe ve složce SampleProducer .

Související úlohy

Spuštění ukázkových aplikací .NET
Ukázkové aplikace .NET můžete spustit interaktivně buď v jednoduchém, nebo rozšířeném režimu, nebo neinteraktivně pomocí automaticky generovaných nebo upravených souborů odpovědí.

Sestavení vlastních aplikací
Sestavujete vlastní aplikace, jako byste sestavili ukázkové aplikace.

“Sestavení vlastních aplikací” na stránce 614
Sestavujete vlastní aplikace, jako byste sestavili ukázkové aplikace.

Sestavení vlastních aplikací

Sestavujete vlastní aplikace, jako byste sestavili ukázkové aplikace.

Než začnete

Nainstalujte příslušný kompilátor. Tato úloha předpokládá, že jste nainstalovali produkt Microsoft Visual Studio 2012 a že jste s jeho použitím obeznámeni.

Procedura

- Sestavte aplikaci .NET , jak je popsáno v tématu “Sestavení ukázkových aplikací .NET” na stránce 614.

  Další pokyny, jak sestavit vlastní aplikace, najdete v souborech Makefile, které jsou k dispozici pro každou ukázkovou aplikaci.

Tip: Chcete-li pomoci s diagnostikou problému v případě selhání, může být užitečné zkompilovat aplikace se zahrnutými symboly.

Související úlohy

Spuštění ukázkových aplikací .NET
Ukázkové aplikace .NET můžete spustit interaktivně buď v jednoduchém, nebo rozšířeném režimu, nebo neinteraktivně pomocí automaticky generovaných nebo upravených souborů odpovědí.

Sestavení ukázkových aplikací .NET
Když sestavíte ukázkovou aplikaci .NET , vytvoří se spustitelná verze vybrané ukázky.

Psaní aplikací XMS

Témata v této sekci poskytují informace, které vám pomohou při psaní aplikací XMS obecně.

Informace o této úloze

Tento oddíl obsahuje obecné koncepty pro psaní aplikací XMS . Informace specifické pro vytváření aplikací XMS .NET naleznete také v tématu “Psaní aplikací XMS .NET” na stránce 632 .

V adresáři IBM MQ 9.2.0se jedná o počet XMS systému XMS.NET byl výrazně snížen na celkem pět. Pět knihoven dynamického propojení:

- `IBM.XMS.dll` -zahrnuje všechny zprávy v národním jazyce
- `IBM.XMS.Comms.RMM.dll`
- Tři knihovny dynamického propojení zásad:
  - `policy.8.0.IBM.XMS.dll`
  - `policy.9.0.IBM.XMS.dll`
  - `policy.9.1.IBM.XMS.dll`

V 9.3.0 Removed V 9.3.0 XMS .NET Výběrové vysílání zpráv (pomocí důvěryhodného výběrového vysílání zpráv) bylo zamítnuto počínaje verzí IBM MQ 9.2 a je odstraněno v IBM MQ 9.3.

Tento oddíl obsahuje následující témata:

- “Model řezání závitů” na stránce 616
- “ConnectionFactories a objekty připojení” na stránce 617
- “Relace” na stránce 618
- “Cíle” na stránce 621
- “Producenti zpráv” na stránce 623
- “Spotřebitelé zpráv” na stránce 624
- “Prohlížeče front” na stránce 627
- “Žadatelé” na stránce 627
- “Odstranění objektu” na stránce 628
- “XMS primitivní typy” na stránce 629
- “Implicitní převod hodnoty vlastnosti z jednoho datového typu na jiný” na stránce 629
- “iterátory” na stránce 631
- “Identifikátory kódované znakové sady” na stránce 632
- “Kódy chyb a výjimek XMS” na stránce 632
- “Sestavení vlastních aplikací” na stránce 614

Windows Použití šablony projektu IBM MQ XMS .NET

Klient produktu IBM MQ XMS .NET vám nabízí možnost používat šablonu projektu, která vám pomůže při vývoji aplikací XMS .NET Core .

Než začnete

V systému musíte mít Microsoft Visual Studio 2017nebo novější a .NET Core 2.1 .

Musíte zkopírovat šablonu XMS .NET z

&MQ_INSTALL_ROOT&\tools\dotnet\samples\cs\core\xms\ProjectTemplates\IBMXMS.NETClientApp.zip

do adresáře

&USER_HOME_DIRECTORY&\Documents\&Visual_Studio_Version&\Templates\ProjectTemplates

adresář, kde:

- *& MQ_INSTALL_ROOT* je kořenový adresář vaší instalace.
- *& USER_HOME_DIRECTORY* je váš domovský adresář.

Chcete-li vyzvednout šablonu, musíte zastavit a restartovat produkt Microsoft Visual Studio .

Informace o této úloze

Šablona projektu XMS .NET obsahuje nějaký obecný kód, který můžete použít k usnadnění vývoje aplikací. Pomocí vestavěného kódu se můžete připojit ke správci front IBM MQ a provést operaci vložení nebo získání prostou úpravou vlastností v vestavěném kódu.

Postup

1. Otevřete produkt Microsoft Visual Studio.
2. Klepněte na volbu **Soubor**, následovanou volbou **Nový** a poté volbou **Projekt**.
3. V *okně Vytvořit nový projekt*vyberte volbu `IBM XMS .NET Client App (.NET Core)` a klepněte na tlačítko **Další**.
4. V okně *Konfigurovat nový projekt* změňte *Název projektu* svého projektu, pokud chcete, a klepnutím na tlačítko **Vytvořit** vytvořte projekt XMS .NET .

   `XMSDotnetApp.cs` je soubor, který je vytvořen spolu se souborem projektu. Tento soubor obsahuje kód, který se připojuje ke správci front, a provádí operaci odeslání a přijetí.

   Vlastnosti připojení jsou nastaveny na výchozí hodnoty:

   - Parametr WMQ_CONNECTION_NAME_LIST je nastaven na hodnotu *localhost (1414)* .
   - XMSC.WMQ_CHANNEL je nastavena na hodnotu *DOTNET.SVRCONN*

   Fronta je nastavena na hodnotu *Q1*a tyto vlastnosti můžete odpovídajícím způsobem upravit.
5. Zkompilujte a spusťte aplikaci.

Související pojmy

Funkce a komponenty IBM MQ

.NET běhové prostředí aplikace- Windows pouze

Model řezání závitů

Obecná pravidla určují, jak může aplikace s podporou podprocesů používat objekty XMS .

- V různých podprocesech lze souběžně používat pouze objekty následujících typů:
  - ConnectionFactory

  - Připojení
  - ConnectionMetaData
  - Místo určení
- Objekt relace lze v daném okamžiku použít pouze pro jeden podproces.

Výjimky z těchto pravidel jsou označeny položkami označenými "Kontext podprocesu" v definicích rozhraní metod v odkazu IBM Message Service Client for .NET.

ConnectionFactories a objekty připojení

Objekt ConnectionFactory poskytuje šablonu, kterou aplikace používá k vytvoření objektu připojení. Aplikace používá objekt připojení k vytvoření objektu relace.

V systému .NETaplikace XMS nejprve používá objekt XMSFactoryFactory k získání odkazu na objekt ConnectionFactory , který odpovídá požadovanému typu protokolu. Tento objekt ConnectionFactory pak může vytvářet připojení pouze pro tento typ protokolu.

Aplikace XMS může vytvořit více připojení a aplikace s podporou podprocesů může používat jeden objekt připojení souběžně ve více podprocesech. Objekt připojení zapouzdřuje komunikační připojení mezi aplikací a serverem systému zpráv.

Spojení slouží k několika účelům:

- Když aplikace vytvoří připojení, lze aplikaci ověřit.
- Aplikace může k připojení přidružit jedinečný identifikátor klienta. Identifikátor klienta se používá pro podporu trvalých odběrů v doméně publikování/odběru. Identifikátor klienta lze nastavit dvěma způsoby:

  Upřednostňovaným způsobem přiřazení identifikátoru klienta připojení je konfigurace v objektu ConnectionFactory specifickém pro klienta s použitím vlastností a transparentním přiřazením k připojení, které vytvoří.

  Alternativním způsobem přiřazení identifikátoru klienta je použití hodnoty specifické pro poskytovatele, která je nastavena na objektu připojení. Tato hodnota nepřepisuje identifikátor, který byl administrativně nakonfigurován. Poskytuje se pro případ, kdy neexistuje žádný administrativně specifikovaný identifikátor. Pokud existuje administrativně zadaný identifikátor, pokus o jeho přepsání hodnotou specifickou pro poskytovatele způsobí vyvolání výjimky. Pokud aplikace explicitně nastaví identifikátor, musí jej provést okamžitě po vytvoření připojení a před provedením dalších akcí na připojení; jinak dojde k výjimce.

Aplikace XMS obvykle vytváří připojení, jednu nebo více relací a několik producentů zpráv a spotřebitelů zpráv.

Vytvoření připojení je relativně nákladné, pokud jde o systémové prostředky, protože zahrnuje vytvoření komunikačního připojení a může také zahrnovat ověření aplikace.

Připojení bylo spuštěno a zastaveno

Připojení může pracovat buď ve spuštěném, nebo zastaveném režimu.

Když aplikace vytvoří připojení, je připojení v zastaveném režimu. Je-li připojení v zastaveném režimu, může aplikace inicializovat relace a může odesílat zprávy, ale nemůže je přijímat, buď synchronně, nebo asynchronně.

Aplikace může spustit připojení voláním metody `Start Connection` . Když je připojení ve spuštěném režimu, může aplikace odesílat a přijímat zprávy. Aplikace pak může zastavit a znovu spustit připojení voláním metod Stop Connection a `Start Connection` .

Uzavření připojení

Aplikace uzavře připojení voláním metody Close Connection. Když aplikace zavře připojení, produkt XMS provede následující akce:

- Zavře všechny relace přidružené k připojení a odstraní určité objekty přidružené k těmto relacím. Další informace o tom, které objekty jsou odstraněny, viz “Odstranění objektu” na stránce 628. Zároveň produkt XMS odvolá všechny aktuálně probíhající transakce v rámci relací.
- Ukončí komunikační spojení se serverem systému zpráv.
- Uvolní paměť a další vnitřní prostředky používané připojením.

Produkt XMS nepotvrdí příjem žádných zpráv, které se nepodařilo potvrdit během relace, před uzavřením připojení. Další informace o potvrzení přijetí zpráv viz “Potvrzení zprávy” na stránce 619.

Zpracování výjimek

Všechny výjimky systému XMS .NET jsou odvozeny od System.Exception. Další informace viz téma “Ošetření chyb v souboru .NET” na stránce 636.

Připojení ke sběrnici pro integraci služeb

Aplikace XMS se může připojit ke sběrnici pro integraci služeb WebSphere Application Server buď pomocí přímého připojení TCP/IP, nebo pomocí HTTP přes TCP/IP.

Protokol HTTP lze použít v situacích, kdy přímé připojení TCP/IP není možné. Jednou z běžných situací je komunikace přes bránu firewall, například když si dva podniky vyměňují zprávy. Použití HTTP ke komunikaci přes bránu firewall se často označuje jako *HTTP tunelové propojení*. Tunelové propojení HTTP je však ze své podstaty pomalejší než použití přímého připojení TCP/IP, protože záhlaví HTTP významně přispívají k množství přenesených dat a protože protokol HTTP vyžaduje více komunikačních toků než TCP/IP.

Chcete-li vytvořit připojení TCP/IP, může aplikace použít továrnu připojení, jejíž vlastnost XMSC_WPM_TARGET_TRANSPORT_CHAIN je nastavena na hodnotu XMSC_WPM_TARGET_TRANSPORT_CHAIN_BASIC. Jedná se o výchozí hodnotu vlastnosti. Pokud je připojení úspěšně vytvořeno, vlastnost XMSC_WPM_CONNECTION_PROTOCOL připojení je nastavena na hodnotu XMSC_WPM_CP_TCP.

Chcete-li vytvořit připojení, které používá protokol HTTP, musí aplikace používat továrnu připojení, jejíž vlastnost XMSC_WPM_TARGET_TRANSPORT_CHAIN je nastavena na název příchozího transportního řetězu, který je konfigurován pro použití kanálu transportu HTTP . Pokud je připojení úspěšně vytvořeno, vlastnost XMSC_WPM_CONNECTION_PROTOCOL připojení je nastavena na hodnotu XMSC_WPM_CP_HTTP. Informace o konfiguraci transportních řetězů naleznete v tématu Konfigurace transportních řetězů v dokumentaci k produktu WebSphere Application Server .

Aplikace má podobný výběr komunikačních protokolů při připojování k serveru samozavedení. Vlastnost XMSC_WPM_PROVIDER_ENDPOINT továrny připojení je posloupností jedné nebo více adres koncových bodů serverů samozavedení. Komponenta transportního řetězu samozavedení každé adresy koncového bodu může být buď XMSC_WPM_BOOTSTRAP_TCP pro připojení TCP/IP k serveru samozavedení, nebo XMSC_WPM_BOOTSTRAP_HTTP pro připojení používající HTTP.

Relace

Relace je kontext s jedním podprocesem pro odesílání a příjem zpráv.

Aplikace může pomocí relace vytvářet zprávy, producenty zpráv, spotřebitele zpráv, prohlížeče front a dočasná místa určení. Aplikace může také použít relaci ke spuštění lokálních transakcí.

Aplikace může vytvořit více relací, kde každá relace vytváří a spotřebovává zprávy nezávisle na ostatních relacích. Pokud se dva spotřebitelé zpráv v oddělených relacích (nebo dokonce ve stejné relaci) přihlásí ke stejnému tématu, obdrží každý kopii jakékoli zprávy publikované v tomto tématu.

Na rozdíl od objektu připojení nelze objekt relace souběžně používat v různých podprocesech. Pouze metodu Close Session objektu Session lze volat z jiného vlákna, než je to, které objekt Session používá v daném okamžiku. Metoda Close Session ukončí relaci a uvolní všechny systémové prostředky přidělené relaci.

Pokud musí aplikace zpracovávat zprávy souběžně ve více než jednom podprocesu, musí vytvořit relaci v každém podprocesu a poté tuto relaci použít pro jakoukoli operaci odeslání nebo přijetí v rámci daného podprocesu.

Transakční relace

Aplikace XMS mohou spouštět lokální transakce. *Lokální transakce* je transakce, která zahrnuje pouze změny prostředků správce front nebo sběrnice SIBus, ke které je aplikace připojena.

Informace v tomto tématu jsou relevantní pouze v případě, že se aplikace připojuje ke správci front IBM MQ nebo ke sběrnici pro integraci služeb WebSphere Application Server . Informace nejsou relevantní pro připojení v reálném čase ke zprostředkovateli.

Chcete-li spustit lokální transakce, musí aplikace nejprve vytvořit relaci transakcí voláním metody Vytvořit relaci objektu připojení a zadat jako parametr, že relace je transakční. Následně jsou všechny zprávy odeslané a přijaté v rámci relace seskupeny do posloupnosti transakcí. Transakce skončí, když aplikace potvrdí nebo odvolá zprávy, které odeslala a přijala od začátku transakce.

Chcete-li potvrdit transakci, aplikace volá metodu Commit objektu Session. Po potvrzení transakce budou všechny zprávy odeslané v rámci transakce k dispozici pro doručení do jiných aplikací a všechny zprávy přijaté v rámci transakce budou potvrzeny tak, aby se server systému zpráv nepokusil o jejich opětovné doručení do aplikace. V doméně typu point-to-point server systému zpráv také odebere přijaté zprávy z jejich front.

Chcete-li odvolat transakci, aplikace volá metodu odvolání objektu relace. Když je transakce odvolána, server systému zpráv zruší všechny zprávy odeslané v rámci transakce a všechny zprávy přijaté v rámci transakce budou znovu k dispozici pro doručení. V doméně typu point-to-point jsou zprávy, které byly přijaty, vloženy zpět do svých front a jsou znovu viditelné pro jiné aplikace.

Nová transakce se spustí automaticky, když aplikace vytvoří relaci transakce nebo zavolá metodu Potvrdit nebo Odvolat. Proto má relace s transakcemi vždy aktivní transakci.

Když aplikace zavře relaci transakce, dojde k implicitnímu odvolání transakce. Když aplikace zavře připojení, dojde k implicitnímu odvolání pro všechny relace transakcí připojení.

Transakce je zcela obsažena v relaci s transakcemi. Transakce nemůže zahrnovat relace. To znamená, že není možné, aby aplikace odesílala a přijímala zprávy ve dvou nebo více relacích transakcí a poté všechny tyto akce potvrzovala nebo odvolala jako jedinou transakci.

Související pojmy

Potvrzení zprávy

Každá relace, která není transakční, má režim potvrzení, který určuje, jak jsou potvrzeny zprávy přijaté aplikací. K dispozici jsou tři režimy potvrzení a volba režimu potvrzení ovlivňuje návrh aplikace.

Doručování zpráv

Produkt XMS podporuje trvalé a přechodné režimy doručování zpráv a asynchronní a synchronní doručování zpráv.

Potvrzení zprávy

Každá relace, která není transakční, má režim potvrzení, který určuje, jak jsou potvrzeny zprávy přijaté aplikací. K dispozici jsou tři režimy potvrzení a volba režimu potvrzení ovlivňuje návrh aplikace.

Informace v tomto tématu jsou relevantní pouze v případě, že se aplikace připojuje ke správci front IBM MQ nebo ke sběrnici pro integraci služeb WebSphere Application Server . Informace nejsou relevantní pro připojení v reálném čase ke zprostředkovateli.

Produkt XMS používá stejný mechanizmus pro potvrzení příjmu zpráv, které používá platforma JMS.

Není-li relace transakční, způsob, jakým jsou zprávy přijaté aplikací potvrzeny, je určen režimem potvrzení relace. Tři režimy potvrzení jsou popsány v následujících odstavcích:

XMSC_AUTO_POTVRZENÍ
Relace automaticky potvrdí každou zprávu přijatou aplikací.

Pokud jsou zprávy doručeny aplikaci synchronně, relace potvrdí přijetí zprávy při každém úspěšném dokončení volání přijetí.

Pokud aplikace úspěšně obdrží zprávu, ale selhání zabrání výskytu potvrzení, bude zpráva znovu k dispozici pro doručení. Aplikace proto musí být schopna zpracovat zprávu, která je znovu doručena.

XMSC_DUPS_OK_POTVRDIT
Relace potvrzuje zprávy přijaté aplikací v době, kdy je vybírá.

Použití tohoto režimu potvrzení snižuje množství práce, kterou musí relace provést, ale selhání, které brání potvrzení zprávy, může vést k opětovnému doručení více než jedné zprávy. Aplikace proto musí být schopna zpracovat zprávy, které jsou znovu doručeny.

XMSC_CLIENT_POTVRDIT
Aplikace potvrdí přijaté zprávy voláním metody Acknowledge třídy Message.

Aplikace může potvrdit příjem každé zprávy jednotlivě, nebo může přijmout dávku zpráv a volat metodu Acknowledge pouze pro poslední zprávu, kterou obdrží. Když je metoda Acknowledge volána, jsou potvrzeny všechny zprávy přijaté od posledního volání metody.

Ve spojení s kterýmkoli z těchto režimů potvrzení může aplikace zastavit a znovu spustit doručování zpráv v relaci voláním metody Recover třídy Session. Zprávy, jejichž příjem byl dříve nepotvrzený, jsou znovu doručeny. Nemusí však být doručeny ve stejném pořadí, v jakém byly doručeny dříve. Mezitím mohly dorazit zprávy s vyšší prioritou a některé původní zprávy mohly vypršet. V doméně typu point-to-point mohly být některé původní zprávy spotřebovány jinou aplikací.

Aplikace může určit, zda je zpráva znovu doručována, kontrolou obsahu pole záhlaví JMSRedelivered zprávy. Aplikace to provede voláním metody Get JMSRedelivered třídy Message.

Související pojmy

Transakční relace
Aplikace XMS mohou spouštět lokální transakce. *Lokální transakce* je transakce, která zahrnuje pouze změny prostředků správce front nebo sběrnice SIBus, ke které je aplikace připojena.

Doručování zpráv
Produkt XMS podporuje trvalé a přechodné režimy doručování zpráv a asynchronní a synchronní doručování zpráv.

Doručování zpráv

Produkt XMS podporuje trvalé a přechodné režimy doručování zpráv a asynchronní a synchronní doručování zpráv.

Režim doručení zpráv

Produkt XMS podporuje dva způsoby doručování zpráv:

Trvalý
Trvalé zprávy jsou doručeny jednou. Server systému zpráv provádí zvláštní opatření, například protokolování zpráv, aby zajistil, že trvalé zprávy nebudou při přenosu ztraceny, a to ani v případě selhání.

Dočasný
Přechodné zprávy jsou doručovány pouze jednou. Přechodné zprávy jsou méně spolehlivé než trvalé zprávy, protože v případě selhání mohou být při přenosu ztraceny.

Volba způsobu doručení je kompromisem mezi spolehlivostí a výkonem. Přechodné zprávy jsou obvykle přenášeny rychleji než trvalé zprávy.

Asynchronní doručování zpráv

Produkt XMS používá jeden podproces ke zpracování všech asynchronních doručení zpráv pro relaci. To znamená, že v daném okamžiku může být spuštěna pouze jedna funkce modulu listener pro zprávy nebo jedna metoda `onMessage()`.

Pokud více než jeden spotřebitel zpráv v relaci přijímá zprávy asynchronně a funkce modulu listener pro zprávy nebo metoda onMessage() doručuje zprávu spotřebiteli zpráv, musí ostatní spotřebitelé zpráv, kteří čekají na stejnou zprávu, pokračovat v čekání. Ostatní zprávy, které čekají na doručení do relace, musí také pokračovat v čekání.

Pokud aplikace vyžaduje souběžné doručení zpráv, vytvořte více než jednu relaci, aby produkt XMS používal k obsluze asynchronního doručování zpráv více než jeden podproces. Tímto způsobem může být souběžně spuštěna více než jedna funkce listeneru zpráv nebo metoda onMessage() .

Relace se nestává asynchronní přiřazením modulu listener pro zprávy ke spotřebiteli. Relace se stane asynchronní pouze při volání metody Connection.Start . Všechna synchronní volání jsou povolena, dokud není volána metoda Connection.Start . Doručení zpráv spotřebitelům začíná při volání funkce Connection.Start .

Pokud musí být v asynchronní relaci provedena synchronní volání, jako např. vytvoření spotřebitele nebo producenta, musí být volána funkce Connection.Stop . Relaci lze obnovit voláním metody Connection.Start , která spustí doručení zpráv. Jedinou výjimkou je podproces doručení zpráv relace, což je podproces, který doručuje zprávy do funkce zpětného volání. Tento podproces může provést libovolné volání relace (kromě volání Close) ve funkci zpětného volání zprávy.

Poznámka: V nespravovaném režimu není volání MQDISC v rámci funkce zpětného volání klientem IBM MQ .NET podporováno. Aplikace klienta tedy nemůže vytvářet nebo zavírat relace v rámci zpětného volání MessageListener v asynchronním režimu příjmu. Vytvořte a zlikvidujte relaci mimo metodu MessageListener .

Synchronní doručování zpráv

Zprávy jsou do aplikace doručovány synchronně, pokud aplikace používá metody Receive objektů MessageConsumer .

Pomocí metod Receive může aplikace čekat určitou dobu na zprávu, nebo může čekat neomezeně dlouho. Případně, pokud aplikace nechce čekat na zprávu, může použít metodu Receive with No Wait.

Související pojmy

Transakční relace
Aplikace XMS mohou spouštět lokální transakce. *Lokální transakce* je transakce, která zahrnuje pouze změny prostředků správce front nebo sběrnice SIBus, ke které je aplikace připojena.

Potvrzení zprávy
Každá relace, která není transakční, má režim potvrzení, který určuje, jak jsou potvrzeny zprávy přijaté aplikací. K dispozici jsou tři režimy potvrzení a volba režimu potvrzení ovlivňuje návrh aplikace.

Cíle

Aplikace XMS používá objekt Destination k určení místa určení odesílaných zpráv a zdroje přijímaných zpráv.

Aplikace XMS může buď vytvořit cílový objekt za běhu, nebo získat předdefinovaný cíl z úložiště spravovaných objektů.

Stejně jako u ConnectionFactoryje nejflexibilnějším způsobem, jak může aplikace XMS určit místo určení, definovat jej jako spravovaný objekt. Pomocí tohoto přístupu mohou aplikace napsané v jazycích C, C + + a .NET a Javasdílet definice místa určení. Vlastnosti spravovaných cílových objektů lze změnit beze změny jakéhokoli kódu.

V případě aplikací .NET vytvoříte cíl pomocí metody CreateTopic nebo CreateQueue . Tyto dvě metody jsou k dispozici v objektech ISession a XMSFactoryFactory v rozhraní API .NET . Další informace naleznete v tématech “Místa určení v .NET” na stránce 634 a ../refdev/sapidest.dita#sapidest.

Identifikátory jednotného prostředku tématu

Identifikátor URI (Uniform Resource Identifier) tématu určuje název tématu; může pro něj také určit jednu nebo více vlastností.

Identifikátor URI pro téma začíná tématem posloupnosti: //, za nímž následuje název tématu a (volitelné) seznam dvojic název-hodnota, které nastavují zbývající vlastnosti tématu. Název tématu nemůže být prázdný.

Zde je příklad ve fragmentu kódu .NET :

```
topic = session.CreateTopic("topic://Sport/Football/Results?multicast=7");
```

Další informace o vlastnostech tématu včetně názvu a platných hodnot, které lze použít v identifikátoru URI, naleznete v tématu Vlastnosti místa určení.

Při zadávání identifikátoru URI tématu pro použití v odběru lze použít zástupné znaky. Syntaxe těchto zástupných znaků závisí na typu připojení a verzi zprostředkovatele. K dispozici je následující volba:

- WebSphere Application Server sběrnice SIBus

WebSphere Application Server sběrnice SIBus

Sběrnice integrace služeb WebSphere Application Server používá následující zástupné znaky:

* pro shodu libovolných znaků na jedné úrovni v hierarchii
// aby odpovídalo 0 nebo více úrovním
// odpovídá 0 nebo více úrovním (na konci výrazu Topic)

Tabulka 82 na stránce 622 uvádí několik příkladů, jak používat toto schéma zástupných znaků.

Tabulka 82. Příklad identifikátorů URI používajících schéma zástupných znaků pro sběrnici pro integraci služeb WebSphere Application Server

Identifikátor jednotného prostředku	**Vyhovuje**	**Příklady**
"aktuální://Sport/ * ball/Výsledky"	Všechna témata s jediným hierarchickým názvem úrovně končícím "míčem" mezi sportem a výsledky	"topic://Sport/Football/Results" a "topic://Sport/Netball/Výsledky"
"aktuální://Sport// Výsledky"	Všechna témata začínající na "Sport/" a končící na "/Výsledky"	"topic://Sport/Football/Results" a "topic://Sport/Hockey/National/Div3/Results"
"topic://Sport/Football//."	Všechna témata začínající na "Sport/Football/"	"topic://Sport/Football/Results" a "topic://Sport/Football/TeamNews/Signings/Managerial"
"topic://Sport/ * ball//Results//."	Témata	"topic://Sport/Football/Results" a "topic://Sport/Netball/National/Div3/Results/2002/November"

Související pojmy

Identifikátory jednotného prostředku fronty
Identifikátor URI pro frontu určuje název fronty; může také určovat jednu nebo více vlastností fronty.

Dočasná místa určení
Aplikace XMS mohou vytvářet a používat dočasné cíle.

Identifikátory jednotného prostředku fronty

Identifikátor URI pro frontu určuje název fronty; může také určovat jednu nebo více vlastností fronty.

Identifikátor URI pro frontu začíná posloupností queue: / /následovanou názvem fronty; může také obsahovat seznam dvojic název-hodnota, které nastavují zbývající vlastnosti fronty.

Pro fronty systému IBM MQ (nikoli však pro výchozí fronty poskytovatele systému zpráv systému WebSphere Application Server) může být správce front, ve kterém je fronta umístěna, zadán před frontou, přičemž název správce front je oddělující od názvu fronty.

Je-li zadán správce front, musí se jednat o správce front, ke kterému je produkt XMS přímo připojen pro připojení pomocí této fronty, nebo musí být přístupný z této fronty. Vzdálení správci front jsou podporováni pouze pro načítání zpráv z front, nikoli pro vkládání zpráv do front. Úplné podrobnosti naleznete v dokumentaci ke správci front IBM MQ .

Není-li určen žádný správce front, je přebytečný oddělovač/oddělovač nepovinný a jeho přítomnost nebo nepřítomnost nijak neliší od definice fronty.

Následující definice front jsou všechny ekvivalentní pro frontu IBM MQ s názvem QB ve správci front s názvem QM_A, ke kterému je produkt XMS přímo připojen:

```
queue://QB
queue:///QB
queue://QM_A/QB
```

Související pojmy

Identifikátory jednotného prostředku tématu
Identifikátor URI (Uniform Resource Identifier) tématu určuje název tématu; může pro něj také určit jednu nebo více vlastností.

Dočasná místa určení
Aplikace XMS mohou vytvářet a používat dočasné cíle.

Dočasná místa určení

Aplikace XMS mohou vytvářet a používat dočasné cíle.

Aplikace obvykle používá dočasné místo určení pro příjem odpovědí na zprávy požadavků. Chcete-li určit místo určení, kam má být odeslána odpověď na zprávu požadavku, aplikace volá metodu Set JMSReplyTo objektu zprávy reprezentující zprávu požadavku. Cíl určený ve volání může být dočasným cílem.

Ačkoli se relace používá k vytvoření dočasného cíle, oborem dočasného cíle je ve skutečnosti připojení, které bylo použito k vytvoření relace. Kterákoli z relací připojení může vytvořit producenty zpráv a spotřebitele zpráv pro dočasné místo určení. Dočasné místo určení zůstane, dokud není explicitně odstraněno, nebo dokud připojení neskončí, podle toho, co nastane dříve.

Když aplikace vytvoří dočasnou frontu, vytvoří se fronta na serveru systému zpráv, ke kterému je aplikace připojena. Pokud je aplikace připojena ke správci front, vytvoří se dynamická fronta z modelové fronty, jejíž název je určen vlastností XMSC_WMQ_TEMPORARY_MODEL a předpona použitá k vytvoření názvu dynamické fronty je určena vlastností XMSC_WMQ_TEMP_Q_PREFIX . Je-li aplikace připojena ke sběrnici pro integraci služeb, vytvoří se ve sběrnici dočasná fronta a předpona, která se používá k vytvoření názvu dočasné fronty, je určena vlastností XMSC_WPM_TEMP_Q_PREFIX .

Když aplikace, která je připojena ke sběrnici pro integraci služeb, vytvoří dočasné téma, je předpona, která se používá k vytvoření názvu dočasného tématu, určena vlastností XMSC_WPM_TEMP_TOPIC_PREFIX .

Související pojmy

Identifikátory jednotného prostředku tématu
Identifikátor URI (Uniform Resource Identifier) tématu určuje název tématu; může pro něj také určit jednu nebo více vlastností.

Identifikátory jednotného prostředku fronty
Identifikátor URI pro frontu určuje název fronty; může také určovat jednu nebo více vlastností fronty.

Producenti zpráv

V produktu XMSlze producent zpráv vytvořit buď s platným místem určení, nebo bez přidruženého místa určení. Při vytváření producenta zpráv s cílem s hodnotou Null je třeba při odesílání zprávy určit platné místo určení.

Producenti zpráv s přidruženým místem určení

V tomto scénáři je producent zpráv vytvořen s použitím platného místa určení. Během operace odeslání není nutné zadat cíl.

Producenti zpráv bez přidruženého místa určení

V produktu XMS .NETlze vytvořit producent zpráv s cílem s hodnotou Null.

Chcete-li vytvořit producenta zpráv bez přidruženého místa určení při použití rozhraní API .NET , musí být hodnota NULL předána jako parametr do metody `CreateProducer()` objektu ISession (například `session.CreateProducer(null)`). Při odesílání zprávy však musí být zadán platný cíl.

Spotřebitelé zpráv

Spotřebitele zpráv lze klasifikovat jako trvalé a dočasné odběratele a synchronní a asynchronní spotřebitele zpráv.

Trvalí odběratelé

Trvalý odběratel je spotřebitel zpráv, který přijímá všechny zprávy publikované v rámci tématu, včetně zpráv publikovaných v době, kdy je odběratel neaktivní.

Informace v tomto tématu jsou relevantní pouze v případě, že se aplikace připojuje ke správci front IBM MQ nebo ke sběrnici pro integraci služeb WebSphere Application Server . Informace nejsou relevantní pro připojení v reálném čase ke zprostředkovateli.

Chcete-li vytvořit trvalého odběratele pro téma, aplikace zavolá metodu Vytvořit trvalého odběratele objektu relace a jako parametry určí název, který identifikuje trvalý odběr, a objekt místa určení, který představuje dané téma. Aplikace může vytvořit trvalého odběratele se selektorem zpráv nebo bez něj a může určit, zda má trvalý odběratel přijímat zprávy publikované vlastním připojením.

Relace použitá k vytvoření trvalého odběratele musí mít přidružený identifikátor klienta. Identifikátor klienta je stejný jako identifikátor přidružený k připojení, které se používá k vytvoření relace; je uveden podle popisu v části “ConnectionFactories a objekty připojení” na stránce 617.

Název identifikující trvalý odběr musí být v rámci identifikátoru klienta jedinečný, a proto je identifikátor klienta součástí úplného jedinečného identifikátoru trvalého odběru. Server systému zpráv udržuje záznam trvalého odběru a zajišťuje, aby všechny zprávy publikované v tématu byly zachovány, dokud nejsou potvrzeny trvalým odběratelem nebo dokud nevyprší jejich platnost.

Server systému zpráv nadále udržuje záznam trvalého odběru i po uzavření trvalého odběratele. Chcete-li znovu použít trvalý odběr, který byl vytvořen dříve, musí aplikace vytvořit trvalého odběratele se stejným názvem odběru a s použitím relace se stejným identifikátorem klienta, který je přidružen k trvalému odběru. Pouze jedna relace v daném okamžiku může mít trvalého odběratele pro konkrétní trvalý odběr.

Oborem trvalého odběru je server systému zpráv, který udržuje záznam o odběru. Pokud dvě aplikace připojené k různým serverům systému zpráv vytvoří trvalého odběratele se stejným názvem odběru a identifikátorem klienta, vytvoří se dva zcela nezávislé trvalé odběry.

Chcete-li odstranit trvalý odběr, aplikace volá metodu zrušení odběru objektu relace a jako parametr určuje název identifikující trvalý odběr. Identifikátor klienta přidružený k relaci musí být stejný jako identifikátor klienta přidružený k trvalému odběru. Server systému zpráv odstraní záznam trvalého odběru, který udržuje, a neodešle žádné další zprávy trvalému odběrateli.

Chcete-li změnit existující odběr, může aplikace vytvořit trvalého odběratele s použitím stejného názvu odběru a identifikátoru klienta, ale s určením jiného tématu nebo selektoru zpráv (nebo obou). Změna trvalého odběru je ekvivalentní odstranění odběru a vytvoření nového.

V případě aplikace, která se připojuje ke správci front IBM MQ , produkt XMS spravuje fronty odběratelů. Aplikace proto nemusí určovat frontu odběratele. XMS bude ignorovat frontu odběratele, je-li uvedena.

Všimněte si, že nemůžete změnit frontu odběratele pro trvalý odběr. Jediným způsobem, jak změnit frontu odběratele, je odstranit odběr a vytvořit nový.

V případě aplikace, která se připojuje ke sběrnici pro integraci služeb, musí mít každý trvalý odběratel určený domovský adresář trvalého odběru. Chcete-li určit domovský adresář trvalého odběru pro všechny trvalé odběratele, kteří používají stejné připojení, nastavte vlastnost XMSC_WPM_DUR_SUB_HOME objektu ConnectionFactory , který se používá k vytvoření připojení. Chcete-li určit domovský adresář trvalého odběru pro konkrétní téma, nastavte vlastnost XMSC_WPM_DUR_SUB_HOME cílového objektu reprezentujícího dané téma. Před vytvořením trvalého odběratele, který používá připojení, musí být pro připojení zadán domovský adresář trvalého odběru. Libovolná hodnota zadaná pro místo určení přepíše hodnotu zadanou pro připojení.

Synchronní a asynchronní spotřebitelé zpráv

Synchronní spotřebitel zpráv přijímá zprávy z fronty synchronně a asynchronní spotřebitel zpráv přijímá zprávy z fronty asynchronně.

Synchronní spotřebitelé zpráv

Synchronní spotřebitel zpráv přijímá vždy jednu zprávu. Při použití metody `Receive(wait interval)` volání čeká na zprávu pouze určenou dobu v milisekundách nebo do zavření spotřebitele zpráv.

Je-li použita metoda ReceiveNoWait (), spotřebitel synchronních zpráv přijímá zprávy bez prodlevy; je-li k dispozici další zpráva, je přijata okamžitě, jinak je vrácen ukazatel na objekt zprávy s hodnotou null.

Asynchronní spotřebitelé zpráv

Modul listener pro zprávy registrovaný aplikací je vyvolán vždy, když je ve frontě k dispozici nová zpráva.

Nezpracovatelné zprávy v adresáři XMS

Nezpracovatelnou zprávou je zpráva, kterou nelze zpracovat přijímající aplikací MDB. Je-li zjištěna nezpracovatelná zpráva, může ji objekt XMS MessageConsumer znovu zobrazit podle dvou vlastností fronty BOQUEUE a BOTHRESH.

Za určitých okolností může být zpráva doručená do objektu MDB odvolána do fronty IBM MQ . K tomu může dojít například v případě, že je zpráva doručena v rámci pracovní jednotky, která je následně odvolána. Zpráva, která je odvolána, je obecně doručena znovu, ale špatně formátovaná zpráva může opakovaně způsobit selhání MDB, a proto nemůže být doručena. Taková zpráva se nazývá nezpracovatelnou zprávou. Produkt IBM MQ můžete nakonfigurovat tak, aby se nezpracovatelné zprávy automaticky přenesly do jiné fronty pro další vyšetřování nebo byly vyřazeny. Chcete-li získat informace o tom, jak nakonfigurovat produkt IBM MQ tímto způsobem, prohlédněte si téma Zpracování nezpracovatelných zpráv v ASF.

Někdy se do fronty dostane nesprávně formátovaná zpráva. V tomto kontextu špatný formát znamená, že přijímající aplikace nemůže zprávu správně zpracovat. Taková zpráva může způsobit selhání přijímající aplikace a vrácení této špatně formátované zprávy zpět. Zpráva pak může být opakovaně doručena do vstupní fronty a opakovaně vrácena aplikací. Tyto zprávy jsou známé jako nezpracovatelné zprávy. Objekt XMS MessageConsumer zjišťuje nezpracovatelné zprávy a směruje je do alternativního místa určení.

Správce front IBM MQ uchovává záznam o počtu případů, kdy byla každá zpráva odvolána. Když toto číslo dosáhne konfigurovatelné prahové hodnoty, spotřebitel zpráv zprávu znovu odešle do pojmenované fronty vrácení. Pokud z nějakého důvodu dojde k selhání tohoto opětného zařazení do fronty, zpráva se odebere ze vstupní fronty a buď se znovu odešle do fronty nedoručených zpráv, nebo se vyřadí.

Objekty XMS ConnectionConsumer zpracovávají nezpracovatelné zprávy stejným způsobem a používají stejné vlastnosti fronty. Pokud více odběratelů připojení monitoruje stejnou frontu, je možné, že nezpracovatelné zprávy mohou být doručeny aplikaci vícekrát, než je prahová hodnota, než dojde k zařazení do fronty. Toto chování je způsobeno způsobem, jakým jednotliví spotřebitelé připojení monitorují fronty a nezpracovatelné zprávy.

Prahová hodnota a název fronty vrácení jsou atributy fronty IBM MQ . Názvy atributů jsou BackoutThreshold a BackoutRequeueQName. Fronta, na kterou se vztahují, je následující:

- Pro systém zpráv typu point-to-point se jedná o základní lokální frontu. To je důležité v případě, že spotřebitelé zpráv a spotřebitelé připojení používají aliasy fronty.

- Pro systém zpráv publikování/odběru v normálním režimu poskytovatele systému zpráv IBM MQ se jedná o modelovou frontu, z níž je vytvořena spravovaná fronta tématu.
- Pro systém zpráv publikování/odběru v režimu migrace poskytovatele systému zpráv IBM MQ se jedná o frontu CCSUB definovanou v objektu továrny TopicConnectionnebo frontu CCDSUB definovanou v objektu Topic.

Chcete-li nastavit atributy QName BackoutThreshold a BackoutRequeue, zadejte následující příkaz MQSC:

```
ALTER QLOCAL(your.queue.name) BOTHRESH(threshold value)
BOQUEUE(your.backout.queue.name)
```

Pro systém zpráv publikování/odběru platí, že pokud váš systém vytvoří dynamickou frontu pro každý odběr, tyto hodnoty atributů jsou získány z modelové fronty IBM MQ classes for JMS , SYSTEM.JMS.MODEL.QUEUE. Chcete-li změnit tato nastavení, použijte:

```
ALTER QMODEL(SYSTEM.JMS.MODEL.QUEUE) BOTHRESH(threshold value)
BOQUEUE(your.backout.queue.name)
```

Je-li prahová hodnota vrácení nula, zpracování nezpracovatelných zpráv je zakázáno a nezpracovatelné zprávy zůstávají ve vstupní frontě. Jinak, když počet vrácení dosáhne prahové hodnoty, zpráva se odešle do pojmenované fronty vrácení.

Pokud počet vrácení dosáhne prahové hodnoty, ale zpráva nemůže přejít do fronty vrácení, zpráva se odešle do fronty nedoručených zpráv, nebo pokud je zpráva dočasná, je vyřazena.

K této situaci dojde, pokud není fronta vrácení definována nebo pokud objekt MessageConsumer nemůže odeslat zprávu do fronty vrácení.

Konfigurace systému pro zpracování nezpracovatelných zpráv

Fronta, kterou produkt XMS .NET používá při zjišťování atributů **BOTHRESH** a **BOQNAME** , závisí na stylu prováděného systému zpráv:

- Pro systém zpráv typu point-to-point se jedná o základní lokální frontu. To je důležité v případě, že aplikace XMS .NET spotřebovává zprávy z alias front nebo z front klastru.
- Pro systém zpráv publikování/odběru se vytvoří spravovaná fronta, která bude uchovávat zprávy pro aplikaci. Produkt XMS .NET se dotazuje spravované fronty a zjišťuje hodnoty atributů **BOTHRESH** a **BOQNAME** .

  Spravovaná fronta je vytvořena z modelové fronty přidružené k objektu Topic, k jehož odběru se aplikace přihlásí, a dědí hodnoty atributů **BOTHRESH** a **BOQNAME** z modelové fronty. Použitá modelová fronta závisí na tom, zda přijímající aplikace vzala trvalý nebo netrvalý odběr:

  – Modelová fronta použitá pro trvalé odběry je určena atributem **MDURMDL** tématu. Výchozí hodnota tohoto atributu je SYSTEM.DURABLE.MODEL.QUEUE.

  – Pro dočasné odběry je použitá modelová fronta určena atributem **MNDURMDL** . Výchozí hodnota atributu **MNDURMDL** je SYSTEM.NDURABLE.MODEL.QUEUE.

Při dotazování na atributy **BOTHRESH** a **BOQNAME** XMS .NET:

- Otevře lokální frontu nebo cílovou frontu pro alias frontu.
- Zjišťuje atributy **BOTHRESH** a **BOQNAME** .
- Zavře lokální frontu nebo cílovou frontu pro alias frontu.

Volby otevření, které se používají při otevírání lokální fronty nebo cílové fronty pro alias fronty, závisí na použité verzi produktu IBM MQ :

- Pro systémy IBM MQ 9.1.0 Fix Pack 4 Long Term Support a dřívější a IBM MQ 9.1.4 Continuous Delivery a dřívější: Je-li lokální fronta nebo cílová fronta pro alias frontu frontou klastru, pak XMS .NET otevře 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 přístup k lokální instanci fronty klastru.

Produkt XMS .NET otevře všechny ostatní typy lokální fronty s volbami otevření MQOO_INQUIRE a MQOO_FAIL_IF_QUIESCING. Aby mohl produkt XMS .NET zadávat dotazy na hodnoty atributů, musí mít uživatel, který spustil přijímající aplikaci, přístup s dotazem na lokální frontu.

- Při použití XMS .NET z IBM MQ 9.1.5 a IBM MQ 9.1.0 Fix Pack 5musí mít uživatel, který spouští přijímající aplikaci, přístup s dotazem na lokální frontu, bez ohledu na typ fronty.

Chcete-li přesunout nezpracovatelné zprávy buď do fronty vrácení, nebo do fronty nedoručených zpráv správce front, musíte udělit uživateli, který spustil aplikaci, oprávnění put a passall .

Zpracování nezpracovatelných zpráv v ASF
Používáte-li funkce ASF (Application Server Facilities), ConnectionConsumernamísto MessageConsumerzpracovává nezpracovatelné zprávy. ConnectionConsumer požaduje zprávy podle vlastností fronty BackoutThreshold a BackoutRequeue.

Když aplikace používá ConnectionConsumers, závisí okolnosti, za kterých je zpráva vrácena, na relaci, kterou poskytuje aplikační server:

- Je-li relace netransakční s parametrem AUTO_KVITOVAT nebo DUPS_OK_KVITOVAT, je zpráva vrácena zpět pouze po chybě systému nebo v případě neočekávaného ukončení aplikace.
- Je-li relace netransakční s CLIENT_ACKNOWLEDGED, mohou být nepotvrzené zprávy zálohovány aplikačním serverem, který volá `Session.recover()`.

  Zpravidla se jedná o implementaci klienta modulu MessageListener nebo o volání aplikačního serveru `Message.acknowledge()`. Produkt `Message.acknowledge()` potvrzuje všechny zprávy dosud doručené v relaci.
- Když je relace transakční, mohou být nepotvrzené zprávy zálohovány aplikačním serverem, který volá `Session.rollback()`.

Prohlížeče front

Aplikace používá prohlížeč front k procházení zpráv ve frontě bez jejich odebrání.

Chcete-li vytvořit prohlížeč front, aplikace zavolá metodu Vytvořit prohlížeč front objektu ISession a jako parametr uvede cílový objekt, který identifikuje frontu, která má být procházena. Aplikace může vytvořit prohlížeč front se selektorem zpráv nebo bez něj.

Po vytvoření prohlížeče front může aplikace volat metodu GetEnumerator objektu IQueueBrowser a získat seznam zpráv ve frontě. Tato metoda vrací výčtový modul, který zapouzdřuje seznam objektů zpráv. Pořadí objektů zpráv v seznamu je stejné jako pořadí, ve kterém budou zprávy načteny z fronty. Aplikace pak může pomocí výčtového nástroje postupně procházet každou zprávu.

Výčtový modul je dynamicky aktualizován, jak jsou zprávy vkládány do fronty a odebírány z fronty. Pokaždé, když aplikace volá IEnumerator.MoveNext() pro procházení další zprávy ve frontě, zpráva odráží aktuální obsah fronty.

Aplikace může pro daný prohlížeč front volat metodu GetEnumerator více než jednou. Každé volání vrátí nový výčtový modul. Aplikace proto může použít více než jeden výčtový modul k procházení zpráv ve frontě a k udržování více pozic ve frontě.

Aplikace může použít prohlížeč front k vyhledání vhodné zprávy k odebrání z fronty a poté použít spotřebitele zpráv s selektorem zpráv k odebrání zprávy. Selektor zpráv může vybrat zprávu podle hodnoty pole záhlaví JMSMessageID . Informace o tomto a dalších polích záhlaví zprávy JMS naleznete v tématu “Pole záhlaví ve zprávě XMS” na stránce 649.

Žadatelé

Aplikace používá žadatele k odeslání zprávy požadavku a poté k čekání na odpověď a k přijetí odpovědi.

Mnoho aplikací systému zpráv je založeno na algoritmech, které odesílají zprávu požadavku a pak čekají na odpověď. Produkt XMS poskytuje třídu s názvem Requestor, která pomáhá s vývojem tohoto stylu aplikace.

Chcete-li vytvořit žadatele, aplikace zavolá konstruktor Vytvořit žadatele třídy žadatele a jako parametry uvede objekt relace a cílový objekt, který identifikuje, kam se mají odeslat zprávy požadavku. Relace nesmí být transakční ani nesmí mít režim potvrzení XMSC_CLIENT_KVITOVAT. Konstruktor automaticky vytvoří dočasnou frontu nebo téma, kam mají být odesílány zprávy s odpovědí.

Po vytvoření žadatele může aplikace volat metodu Request objektu Requestor, aby odeslala zprávu požadavku a pak počkala na a přijala odpověď od aplikace, která přijala zprávu požadavku. Volání čeká na přijetí odpovědi nebo na ukončení relace, podle toho, co nastane dříve. Žadatel požaduje pouze jednu odpověď pro každou zprávu požadavku.

Když aplikace zavře žadatele, dočasná fronta nebo téma se odstraní. Přidružená relace se však nezavře.

Odstranění objektu

Když aplikace odstraní objekt XMS , který vytvořila, produkt XMS uvolní vnitřní prostředky, které byly k objektu přiděleny.

Když aplikace vytvoří objekt XMS , XMS přidělí objektu paměť a další vnitřní prostředky. Produkt XMS tyto interní prostředky zachová, dokud aplikace explicitně neodstraní objekt voláním metody close nebo delete objektu. V tomto okamžiku produkt XMS uvolní interní prostředky. Pokud se aplikace pokusí odstranit objekt, který je již odstraněn, volání se ignoruje.

Když aplikace odstraní objekt připojení nebo relace, produkt XMS automaticky odstraní určité přidružené objekty a uvolní jejich vnitřní prostředky. Jedná se o objekty, které byly vytvořeny objektem Připojení nebo Relace a nemají žádnou funkci nezávislou na objektu. Tyto objekty jsou zobrazeny v souboru Tabulka 83 na stránce 628.

Poznámka: Pokud aplikace zavře připojení se závislými relacemi, všechny objekty závislé na těchto relacích se také odstraní. Pouze objekt Připojení nebo Relace může mít závislé objekty.

Tabulka 83. Objekty, které jsou automaticky odstraněny

Odstraněný objekt	**Metoda**	**Závislé objekty, které jsou automaticky odstraněny**
Připojení	Zavřít připojení	ConnectionMetaDatové a relační objekty
Relace	Zavřít relaci	MessageConsumer, MessageProducer, QueueBrowsera objekty Requestor.

Spravované IBM MQ transakce XA prostřednictvím XMS

Spravované transakce IBM MQ XA lze používat prostřednictvím produktu XMS.

Chcete-li používat transakce XA prostřednictvím produktu XMS, je třeba vytvořit relaci transakcí. Když je transakce XA používána, je řízení transakcí prostřednictvím globálních transakcí DTC (Distributed Transaction Coordinator) a není to relace XMS . Při použití transakcí XA nelze v relaci XMS zadat příkaz `Session.commit` nebo `Session.rollback` . Místo toho použijte metody `Transscope.Commit` nebo `Transscope.Rollback` DTC potvrzení nebo odvolání transakcí. Je-li pro transakci XA použita relace, musí být součástí transakce XA producent nebo spotřebitel vytvořený pomocí této relace. Nelze je použít pro žádné operace mimo rozsah transakce XA. Nelze je použít pro operace jako `Producer.send` nebo `Consumer.receive` mimo transakci XA.

Objekt výjimky `IllegalStateException` je vyvolán, pokud:

- Transakční relace XA se používá pro `Session.commit` nebo `Session.rollback`.
- Objekty producenta nebo spotřebitele, které jsou jednou použity v transakční relaci XA, jsou použity mimo obor transakce XA.

Transakce XA nejsou podporovány v asynchronních spotřebitelích.

Poznámka:

1. Před potvrzením transakce XA může být na objektu `Producer`, `Consumer`, `Session`nebo `Connection` vydáno zavření. Ve kterých případech jsou zprávy v transakci odvolány. Podobně, pokud

je připojení přerušeno před potvrzením transakce XA, všechny zprávy v transakci jsou odvolány. Pro objekt `Producer` odvolání znamená, že zprávy nejsou vloženy do fronty. Pro objekt `Consumer` odvolání znamená, že zprávy zůstávají ve frontě.

2. Pokud objekt `Producer` vloží zprávu s hodnotou `TimeToLive` do souboru `TransactionScope` a po uplynutí času se vydá zpráva `commit` , může tato zpráva vypršet před vydáním příkazu `commit` . V tomto případě není zpráva zpřístupněna pro objekty `Consumer` .
3. Objekty `Session` nejsou podporovány v rámci podprocesů. Použití transakcí s objekty `Session` , které jsou sdíleny mezi podprocesy, není podporováno.

XMS primitivní typy

XMS poskytuje ekvivalenty osmi Java primitivních typů (byte, short, int, long, float, double, char a boolean). To umožňuje výměnu zpráv mezi XMS a JMS bez ztráty nebo poškození dat.

Tabulka 84 na stránce 629 uvádí Java ekvivalentní datový typ, velikost a minimální a maximální hodnotu každého XMS primitivního typu.

Tabulka 84. XMS datové typy a jejich Java ekvivalenty

Datový typ XMS	Kompatibilní datový typJava	Velikost	Minimální hodnota	Maximální hodnota
System.Boolean	typ boolean	32 bitů	ne	ano
System.SBYTE	bajt	8 bitů	-2^7 (-128)	2^7-1 (127)
System.BYTE	bajt	8 bitů	-2^7 (-128)	2^7-1 (127)
System.CHAR	bajt	8 bitů	-2^7 (-128)	2^7-1 (127)
System.Int16	short	16 bitů	-2^{15} (-32768)	2^{15}-1 (32767)
System.Int32	celé číslo	32 bitů	-2^{31} (-2147483648)	2^{31}-1 (2147483647)
System.Int64	long	64 bitů	-2^{63} (-9223372036854775808)	2^{63}-1 (9223372036854775807)
System.Single	float	32 bitů	-3.402823E-38 (s přesností na 7 číslic)	3.402823E+38 (s přesností na 7 číslic)
System.Double	dvojitý	64 bitů	-1.79769313486231E-308 (s přesností na 15 číslic)	1.79769313486231E+308 (s přesností na 15 číslic)

Implicitní převod hodnoty vlastnosti z jednoho datového typu na jiný

Když aplikace získá hodnotu vlastnosti, může ji produkt XMS převést na jiný datový typ. Mnoho pravidel řídí, které převody jsou podporovány a jak XMS provádí převody.

Vlastnost objektu má název a hodnotu; hodnota má přidružený datový typ, kde se hodnota vlastnosti také označuje jako *typ vlastnosti*.

Aplikace používá metody třídy PropertyContext k získání a nastavení vlastností objektů. Chcete-li získat hodnotu vlastnosti, aplikace volá metodu, která je vhodná pro daný typ vlastnosti. Chcete-li například získat hodnotu celočíselné vlastnosti, aplikace obvykle volá metodu GetIntProperty.

Když však aplikace získá hodnotu vlastnosti, může ji produkt XMS převést na jiný datový typ. Chcete-li například získat hodnotu celočíselné vlastnosti, může aplikace volat metodu GetStringProperty, která vrací hodnotu vlastnosti jako řetězec. Převody podporované produktem XMS jsou uvedeny v souboru Tabulka 85 na stránce 630.

Tabulka 85. Podporované převody z typu vlastnosti na jiné datové typy	
Typ vlastnosti	**Podporované cílové datové typy**
System.String	System.Boolean, System.Double, System.Float, System.Int32, System.Int64, System.SByte, System.Int16
System.Boolean	System.String, System.SByte, System.Int32, System.Int64, System.Int16
System.Char	System.String
System.Double	System.String
System.Float	System.String, System.Double
System.Int32	System.String, System.Int64
System.Int64	System.String
System.SByte	System.String, System.Int32, System.Int64, System.Int16
Pole System.SByte	System.String
System.Int16	System.String, System.Int32, System.Int64

Podporované konverze se řídí následujícími obecnými pravidly:

- Číselné hodnoty vlastností lze převést z jednoho datového typu na jiný za předpokladu, že během převodu nebudou ztracena žádná data. Například hodnotu vlastnosti s datovým typem System.Int32 lze převést na hodnotu s datovým typem System.Int64, ale nelze ji převést na hodnotu s datovým typem System.Int16.
- Hodnotu vlastnosti libovolného datového typu lze převést na řetězec.
- Hodnotu vlastnosti řetězce lze převést na jakýkoli jiný datový typ za předpokladu, že je řetězec pro převod správně naformátován. Pokud se aplikace pokusí převést hodnotu vlastnosti řetězce, která není správně naformátována, produkt XMS může vrátit chyby.
- Pokud se aplikace pokusí o převod, který není podporován, produkt XMS může vrátit chybu.

Při převodu hodnoty vlastnosti z jednoho datového typu na jiný se použijí následující pravidla:

- Při převodu hodnoty logické vlastnosti na řetězec se hodnota true převede na řetězec "true"a hodnota false se převede na řetězec "false".
- Při převodu hodnoty logické vlastnosti na číselný datový typ, včetně System.SByte, se hodnota true převede na 1 a hodnota false se převede na 0.
- Při převodu hodnoty vlastnosti řetězce na logickou hodnotu je řetězec "true" (bez rozlišování velkých a malých písmen) nebo "1" převeden na hodnotu true a řetězec "false" (bez rozlišování velkých a malých písmen) nebo "0" je převeden na hodnotu false. Všechny ostatní řetězce nelze převést.
- Při převodu hodnoty vlastnosti řetězce na hodnotu s datovým typem System.Int32, System.Int64, System.SBytenebo System.Int16musí mít řetězec následující formát:

  [*mezery*] [*znak*]*číslice*

  Řetězcové komponenty jsou definovány takto:

  mezery
  Volitelné úvodní mezery.

  SIGN
  Volitelný znak plus (+) nebo znak minus (-).

  číslice
  Souvislá posloupnost číselných znaků (0-9). Musí být uveden alespoň jeden znak číslice.

  Po posloupnosti číselných znaků může řetězec obsahovat další znaky, které nejsou číselnými znaky, ale převod se zastaví, jakmile je dosaženo prvního z těchto znaků. Předpokládá se, že řetězec představuje desetinné celé číslo.

XMS může vrátit chybu, pokud není řetězec správně naformátován.

- Při převodu hodnoty vlastnosti řetězce na hodnotu s datovým typem System.Double nebo System.Floatmusí mít řetězec následující formát:

  [*mezery*] [*znak*] [*číslice*] [*bod*[*d_číslice*]] [*e_char*[*e_sign*]*e_číslice*]

  Řetězcové komponenty jsou definovány takto:

  mezery
  (Volitelné) Vedoucí prázdné znaky.

  SIGN
  (Volitelné) Znak plus (+) nebo znak minus (-).

  číslice
  Souvislá posloupnost číselných znaků (0-9). Alespoň jeden znak číslice musí být přítomen buď v poli *číslice* , nebo v poli *d_číslice*.

  bod
  (Nepovinné) desetinný bod (.).

  d_číslice
  Souvislá posloupnost číselných znaků (0-9). Alespoň jeden znak číslice musí být přítomen buď v poli *číslice* , nebo v poli *d_číslice*.

  e_char
  Exponent, který je buď *E* , nebo *e*.

  e_sign
  (Volitelné) Znak plus (+) nebo znak minus (-) pro exponent.

  e_číslice
  Souvislá posloupnost číselných znaků (0-9) pro exponent. Pokud řetězec obsahuje znak exponentu, musí být uveden alespoň jeden znak číslice.

  Po posloupnosti číselných znaků nebo volitelných znaků reprezentujících exponent může řetězec obsahovat jiné znaky, které nejsou číselnými znaky, ale převod se zastaví, jakmile je dosaženo prvního z těchto znaků. Předpokládá se, že řetězec představuje desetinné číslo s pohyblivou řádovou čárkou s exponentem, který je mocninou 10.

  XMS může vrátit chybu, pokud není řetězec správně naformátován.

- Při převodu hodnoty číselné vlastnosti na řetězec, včetně hodnoty vlastnosti s datovým typem System.SByte, je hodnota převedena na řetězcovou reprezentaci hodnoty jako desetinné číslo, nikoli řetězec obsahující pro tuto hodnotu znak ASCII. Například celé číslo 65 se převede na řetězec "65", nikoli na řetězec "A".
- Při převodu hodnoty vlastnosti bajtového pole na řetězec je každý bajt převeden na 2 hexadecimální znaky, které představují bajt. Například bajtové pole {0xF1, 0x12, 0x00, 0xFF} je převedeno na řetězec "F11200FF".

Převody z typu vlastnosti na jiné datové typy jsou podporovány metodami tříd Property a PropertyContext .

iterátory

Iterátor zapouzdřuje seznam objektů a kurzor, který udržuje aktuální pozici v seznamu. Koncept iterátoru, který je k dispozici v produktu IBM MQ Message Service Client (XMS) for C/C++, je implementován pomocí rozhraní IEnumerator v produktu IBM MQ Message Service Client (XMS) for .NET.

Při vytvoření iterátoru je pozice kurzoru před prvním objektem. Aplikace používá iterátor k načtení jednotlivých objektů.

Třída iterátoru IBM MQ Message Service Client (XMS) for C/C++ je ekvivalentní třídě výčtového modulu v souboru Java. IBM MQ Message Service Client (XMS) for .NET je podobný Java a používá rozhraní IEnumerator.

Aplikace může použít výčet IEnumerator k provedení následujících úloh:

- Získání vlastností zprávy
- Získání dvojic název-hodnota v těle zprávy mapy
- Procházení zpráv ve frontě
- Chcete-li získat názvy vlastností zpráv definovaných v souboru JMS , které jsou podporovány připojením,

Identifikátory kódované znakové sady

V produktu XMS .NETse všechny řetězce předávají pomocí nativního řetězce .NET . Vzhledem k tomu, že má pevné kódování, nejsou k jeho interpretaci zapotřebí žádné další informace. Proto není vlastnost XMSC_CLIENT_CCSID požadována pro aplikace XMS .NET .

Kódy chyb a výjimek XMS

Produkt XMS používá k označení selhání rozsah kódů chyb. Tyto kódy chyb nejsou v této dokumentaci výslovně uvedeny, protože se mohou v jednotlivých verzích lišit. Dokumentovány jsou pouze kódy výjimek XMS (ve formátu XMS_X_ ...), protože zůstávají stejné ve všech vydáních produktu XMS.

Automatické IBM MQ opětovné připojení klienta prostřednictvím XMS

Konfigurujte klienta XMS tak, aby se automaticky znovu připojoval po selhání sítě, správce front nebo serveru při použití klienta IBM WebSphere MQ 7.1 a novějšího jako poskytovatele zpráv.

Pomocí vlastností `WMQ_CONNECTION_NAME_LIST` a `WMQ_CLIENT_RECONNECT_OPTIONS` třídy `MQConnectionFactory` nakonfigurujte připojení klienta pro automatické opětovné připojení. Automatické opětovné připojení klienta znovu připojí klienta po selhání připojení nebo jako volbu po zastavení správce front. Návrh některých klientských aplikací je činí nevhodnými pro automatické opětovné připojení.

Automaticky znovu připojitelná klientská připojení se stanou znovu připojitelná po navázání připojení.

Poznámka: Vlastnosti `Volby opětovného připojení klienta`, `Časový limit opětovného připojení klientaa Seznam názvů připojení` lze nastavit také prostřednictvím tabulky CCDT (Client Channel Definitions) nebo povolením opětovného připojení klienta prostřednictvím souboru `mqclient.ini` .

Poznámka: Jsou-li pro objekt `ConnectionFactory` a také v tabulce CCDT nastaveny vlastnosti opětovného připojení, je pravidlo priority následující. Je-li v objektu `ConnectionFactory` nastavena výchozí hodnota vlastnosti seznamu názvů připojení, má přednost tabulka CCDT. Není-li seznam názvů připojení nastaven na výchozí hodnotu, mají přednost hodnoty vlastností nastavené v objektu `ConnectionFactory` . Výchozí hodnota seznamu názvů připojení je `localhost(1414)`.

Psaní aplikací XMS .NET

Témata v této sekci poskytují informace, které vám pomohou při psaní aplikací XMS .NET .

Informace o této úloze

Tento oddíl poskytuje informace, které jsou specifické pro psaní aplikací XMS .NET . Obecné informace o psaní aplikací XMS viz “Psaní aplikací XMS” na stránce 615.

Tento oddíl obsahuje následující témata:

Datové typy pro .NET

XMS .NET podporuje System.Boolean, System.Byte, System.SByte, System.Char, System.String, System.Single, System.Double, System.Decimal, System.Int16, System.Int32, System.Int64, System.UInt16, System.UInt32, System.UInt64, a System.Object. Datové typy pro produkt XMS .NET se liší od datových typů pro produkt XMS C/C + +. Toto téma můžete použít k identifikaci odpovídajících datových typů.

Následující tabulka zobrazuje odpovídající datové typy XMS .NET a XMS C/C++ a stručně je popisuje.

Tabulka 86. Datové typy pro XMS .NET a XMS C/C++

XMS .NET typ	**XMS Typ C/C++**	**Popis**
System.SByte	xmsSBYTE xmsINT8	Podepsaná 8bitová hodnota
System.Byte	xmsBYTE xmsUINT8	8bitová hodnota bez znaménka
System.Int16	xmsINT16 xmsSHORT	16bitová hodnota se znaménkem
System.UInt16	xmsUINT16 xmsUSHORT	16bitová hodnota bez znaménka
System.Int32	xmsINT32 xmsINT	32bitová hodnota se znaménkem
System.UInt32	xmsUINT32 xmsUINT	32bitová hodnota bez znaménka
System.Int64	xmsLONG xmsINT64	64bitová hodnota se znaménkem
System.UInt64	xmsULONG xmsUINT64	64bitová hodnota bez znaménka
System.Char	xmsCHAR16	16bitový znak bez znaménka (Unicode pro .NET)
System.Single	xmsFLOAT	IEEE 32bitový typ float
System.Double	xmsDOUBLE	IEEE 64bitový typ float
System.Boolean	xmsBOOL	Hodnota True/False
Nelze použít	xmsCHAR	Podepsaná nebo nepodepsaná 8bitová hodnota (podepsaná nebo nepodepsaná závisí na platformě)
System.Decimal	Nelze použít	96bitové celé číslo se znaménkem 10^0 až 10^{28}
System.Object	Nelze použít	Základ všech typů
System.String	Nelze použít	Řetězcový typ

Spravované a nespravované operace v produktu .NET

Spravovaný kód je spouštěn výhradně v běhovém prostředí společného jazyka .NET a je zcela závislý na službách poskytovaných tímto běhovým prostředím. Aplikace je klasifikována jako nespravovaná, pokud je jakákoli část aplikace spuštěna nebo volá služby mimo běhové prostředí společného jazyka .NET .

Některé rozšířené funkce nelze v současné době ve spravovaném prostředí .NET podporovat.

Pokud vaše aplikace vyžaduje některé funkce, které nejsou v současné době podporovány v plně spravovaném prostředí, můžete aplikaci změnit tak, aby používala nespravované prostředí, aniž byste museli aplikaci podstatně změnit. Měli byste si však uvědomit, že zásobník XMS používá při výběru nespravovaný kód.

Připojení ke správci front IBM MQ

Spravovaná připojení k WMQ_CM_CLIENT nebudou podporovat jiné komunikace než TCP a kompresi kanálu. Tato připojení však mohou být podporována pomocí nespravovaného připojení (WMQ_CM_CLIENT_UNMANAGED). Další informace viz téma “Vývoj aplikací .NET” na stránce 539.

Pokud vytvoříte továrnu připojení ze spravovaného objektu v nespravovaném prostředí, musíte ručně změnit hodnotu pro režim připojení na XMSC_WMQ_CM_CLIENT_UNMANAGED.

Připojení ke stroji systému zpráv sběrnice pro integraci služeb WebSphere Application Server

Připojení ke stroji systému zpráv sběrnice SIBus systému WebSphere Application Server , která vyžadují použití protokolu SSL (včetně protokolu HTTPS), nejsou v současné době podporována jako spravovaný kód.

Místa určení v .NET

V produktu .NETjsou cíle vytvořeny podle typu protokolu a lze je použít pouze pro typ protokolu, pro který jsou vytvořeny.

Pro vytváření cílů jsou k dispozici dvě funkce, jedna pro témata a jedna pro fronty:

- `IDestination CreateTopic(String topic);`
- `IDestination CreateQueue(String queue);`

Tyto funkce jsou k dispozici na následujících dvou objektech v rozhraní API:

- IRelace
- XMSFactoryFactory

V obou případech mohou tyto metody přijmout řetězec stylu identifikátoru URI, který může obsahovat parametry, v následujícím formátu:

```
"topic://some/topic/name?priority=5"
```

Alternativně mohou tyto metody přijmout pouze název místa určení, tj. název bez předpony topic: // nebo queue: // a bez parametrů.

To znamená, že následující řetězec stylu identifikátoru URI:

```
CreateTopic("topic://some/topic/name");
```

by vedlo ke stejnému výsledku jako následující název místa určení:

```
CreateTopic("some/topic/name");
```

Stejně jako u WebSphere Application Server sběrnice SIBus JMSlze témata zadat také ve zkráceném formátu, který zahrnuje jak *název_tématu* , tak *prostor_tématu* , ale nemůže obsahovat parametry:

```
CreateTopic("topicspace:topicname");
```

Vlastnosti v souboru .NET

Aplikace .NET používá metody v rozhraní PropertyContext k získání a nastavení vlastností objektů.

Rozhraní PropertyContext zapouzdřuje metody, které získají a nastaví vlastnosti. Tyto metody jsou přímo či nepřímo zděděny následujícími třídami:

- BytesMessage
- Připojení
- ConnectionFactory
- ConnectionMetaData
- Místo určení
- MapMessage
- Zpráva
- MessageConsumer
- MessageProducer
- ObjectMessage
- QueueBrowser
- Relace
- StreamMessage
- TextMessage

Pokud aplikace nastaví hodnotu vlastnosti, nová hodnota nahradí všechny předchozí hodnoty, které tato vlastnost měla.

Další informace o vlastnostech XMS naleznete v tématu Vlastnosti objektů XMS.

Pro snadné použití jsou názvy a hodnoty vlastností XMS v produktu XMS předdefinovány jako veřejné konstanty ve struktuře s názvem XMSC. Názvy těchto konstant jsou ve formátu XMSC.*konstanta*; například XMSC.USERID (konstanta názvu vlastnosti) a XMSC.DELIVERY_AS_APP (hodnotová konstanta).

Dále můžete přistupovat k konstantám IBM MQ pomocí IBM.XMS.MQC . Pokud je IBM.XMS je již naimportován. K hodnotám těchto vlastností můžete přistupovat ve formátu MQC.*constant*. Například MQC.MQRO_COA_WITH_FULL_DATA.

Dále, pokud máte hybridní aplikaci, která používá třídy XMS .NET i IBM MQ pro .NET a která importuje obě IBM.XMS a IBM.WMQ pak musíte plně kvalifikovat obor názvů struktury MQC, abyste se ujistili, že je každý výskyt jedinečný.

Některé rozšířené funkce nejsou v současné době podporovány ve spravovaném prostředí .NET . Další podrobnosti viz “Spravované a nespravované operace v produktu .NET” na stránce 634 .

Neexistující zpracování vlastností v souboru .NET

Zpracování neexistujících vlastností v produktu XMS .NET je obecně konzistentní se specifikací JMS a také s implementacemi jazyka C a C++ produktu XMS.

V prostředí JMS může přístup k neexistující vlastnosti vést k výjimce systému Java , když se metoda pokusí převést neexistující hodnotu (null) na požadovaný typ. Pokud vlastnost neexistuje, dojde k následujícím výjimkám:

- getStringVlastnost a vlastnost getObjectvrací hodnotu null

- getBooleanVlastnost vrací false, protože Boolean.valueOf(null) vrací false
- getIntVlastnost etc.throw java.lang.NumberFormatException , protože Integer.valueOf(null) vygeneruje výjimku

Pokud vlastnost v produktu XMS .NET neexistuje, dojde k následujícím výjimkám:

- Funkce GetStringProperty a GetObjectProperty (a GetBytesProperty) vrací hodnotu null (což je stejné jako Java).
- GetBooleanvyvolá vlastnost System.NullReferenceException
- GetIntVlastnost atd. vyvolá výjimku System.NullReferenceException

Tato implementace se liší od implementace Java, ale je široce konzistentní se specifikací JMS a s rozhraními XMS C a C + +. Podobně jako implementace Java produkt XMS .NET šíří všechny výjimky z volání System.Convert volajícímu. Na rozdíl od Java však XMS explicitně generuje výjimky NullReference, spíše než aby používala nativní chování rámce .NET prostřednictvím předávání hodnot null do rutin pro převod systému. Pokud aplikace nastaví vlastnost na řetězec jako "abc" a zavolá vlastnost GetInt, dojde k výjimce System.FormatException vyvolané příkazem Convert.ToInt32("abc") je šířen na volajícího, což je konzistentní s Java. MessageFormatVýjimka je vyvolána pouze v případě, že typy použité pro setProperty a getProperty jsou nekompatibilní. Toto chování je také konzistentní s Java.

Ošetření chyb v souboru .NET

Všechny výjimky systému XMS .NET jsou odvozeny od System.Exception. Volání metody XMS může vyvolat specifické výjimky XMS , jako např. výjimku MessageFormat, obecnou výjimku XMSExceptions nebo systémové výjimky, jako např. NullReference.

Zapište aplikace, aby zachytily jakoukoli z těchto chyb, buď ve specifických blocích catch, nebo v obecných blocích catch `System.Exception` , podle požadavků aplikací.

Použití modulů listener pro zprávy a výjimky v adresáři .NET

Aplikace .NET používá modul listener pro zprávy k asynchronnímu příjmu zpráv a používá modul listener pro výjimky k asynchronnímu upozornění na problém s připojením.

Informace o této úloze

Funkčnost modulů listener pro zprávy i výjimky je stejná pro produkt .NET i pro jazyk C + +. Existují však některé malé rozdíly v implementaci.

Procedura

- Chcete-li nastavit modul listener pro zprávy tak, aby přijímal zprávy asynchronně, postupujte takto:
  a) Definujte metodu, která odpovídá podpisu delegáta modulu listener pro zprávy.
  Metoda, kterou definujete, může být buď statická, nebo instanční metoda a může být definována v libovolné přístupné třídě. Podpis delegáta je následující:

```
public delegate void MessageListener(IMessage msg);
```

  a tak jste mohli definovat metodu jako:

```
void SomeMethodName(IMessage msg);
```

  b) Vytvořte instanci této metody jako delegáta pomocí něčeho podobného následujícímu příkladu:

```
MessageListener OnMsgMethod = new MessageListener(SomeMethodName)
```

  c) Zaregistrujte delegáta u jednoho nebo více spotřebitelů jeho nastavením na vlastnost MessageListener spotřebitele:

```
consumer.MessageListener = OnMsgMethod;
```

Delegáta můžete odebrat nastavením MessageListener zpět na hodnotu null:

```
consumer.MessageListener = null;
```

- Chcete-li nastavit modul listener pro výjimky, postupujte takto.

  Modul listener pro výjimky pracuje v podstatě stejně jako modul listener pro zprávy, ale má jinou definici delegáta a je přiřazen k připojení, spíše než ke spotřebiteli zpráv. Toto je stejné jako pro C + +.

  a) Definujte metodu.

  Podpis delegáta je následující:

```
public delegate void ExceptionListener(Exception ex);
```

  a tak by metoda mohla být definována:

```
void SomeMethodName(Exception ex);
```

  b) Vytvořte instanci této metody jako delegáta pomocí něčeho podobného následujícímu příkladu:

```
ExceptionListener OnExMethod = new ExceptionListener(SomeMethodName)
```

  c) Registrujte delegáta s připojením nastavením jeho vlastnosti ExceptionListener :

```
connection.ExceptionListener = OnExMethod ;
```

  Delegáta můžete odebrat resetováním ExceptionListener na:

```
null:  connection.ExceptionListener = null;
```

Práce se spravovanými objekty XMS .NET

Témata v této sekci poskytují informace o spravovaných objektech. Aplikace systému XMS mohou načítat definice objektů z centrálního úložiště spravovaných objektů a používat je k vytváření továren připojení a míst určení.

Informace o této úloze

Tento oddíl poskytuje informace, které vám pomohou s vytvářením a správou spravovaných objektů, a popisuje typy spravovaného úložiště objektů, které produkt XMS podporuje. Tento oddíl také vysvětluje, jak aplikace XMS vytváří připojení k úložišti spravovaných objektů za účelem načtení požadovaných spravovaných objektů.

Tento oddíl obsahuje následující témata:

- “Webová služba vyhledání v rozhraní JNDI pro XMS .NET” na stránce 644
- “XMS .NET načtení spravovaných objektů” na stránce 645

XMS .NET podporované typy úložiště spravovaných objektů

Spravované objekty systému souborů a LDAP lze použít pro připojení k IBM MQ a WebSphere Application Server, zatímco pojmenování COS lze použít pouze pro připojení k serveru WebSphere Application Server.

Adresáře objektů systému souborů mají formu serializovaných objektů Java Naming Directory Interface (JNDI). Adresáře objektů LDAP jsou adresáře, které obsahují objekty JNDI . Adresáře objektů systému souborů a LDAP mohou být spravovány pomocí IBM MQ Explorer, který je poskytován s produktem IBM MQ a novějším. Systém souborů i adresáře objektů LDAP lze použít ke správě připojení klienta prostřednictvím centralizace továren připojení a míst určení produktu IBM MQ . Administrátor sítě může implementovat více aplikací, které odkazují na stejné centrální úložiště, a které jsou automaticky aktualizovány tak, aby odrážely změny nastavení připojení provedené v centrálním úložišti.

Adresář pojmenování COS obsahuje továrny připojení a místa určení WebSphere Application Server service integration bus a lze jej spravovat pomocí administrativní konzoly produktu WebSphere Application Server . Aby mohla aplikace XMS načíst objekty z adresáře pojmenování COS, musí být implementována vyhledávací webová služba JNDI . Tato webová služba není k dispozici na všech WebSphere Application Server service integration technologies. Podrobnosti naleznete v dokumentaci k produktu.

Poznámka: Restartujte připojení aplikace, aby se změny v adresáři objektů projevily.

Mapování vlastností XMS .NET pro spravované objekty

Chcete-li povolit aplikacím XMS .NET používat definice továrny připojení a cílového objektu produktu IBM MQ JMS a WebSphere Application Server , musí být vlastnosti načtené z těchto definic mapovány na odpovídající vlastnosti produktu XMS , které lze nastavit v továrnách a cílech připojení XMS .

Chcete-li například vytvořit továrnu připojení produktu XMS s vlastnostmi, které jsou načteny z továrny připojení produktu IBM MQ JMS, musí být vlastnosti mapovány mezi těmito dvěma vlastnostmi.

Všechna mapování vlastností se provádějí automaticky.

Tabulka 87 na stránce 638 demonstruje mapování mezi některými z nejběžnějších vlastností továren připojení a míst určení. Vlastnosti zobrazené v této tabulce jsou jen malou sadou příkladů a ne všechny zobrazené vlastnosti jsou relevantní pro všechny typy připojení a servery.

Tabulka 87. Příklady mapování názvů pro továrnu připojení a vlastnosti místa určení

IBM MQ JMS název vlastnosti	**XMS název vlastnosti**	**WebSphere Application Server service integration bus název vlastnosti**
PERZISTENCE (ZA)	XMSC_DELIVERY_MODE	
UKONČENÍ PLATNOSTI (EXP)	XMSC_TIME_TO_LIVE	
PRIORITA (PRI)	XMSC_PRIORITY	
	XMSC_WPM_HOST_NAME	serverName
	XMSC_WPM_BUS_NAME	busName
	XMSC_WPM_TOPIC_SPACE	topicName

Poznámka: Vlastnosti uvedené v části Tabulka 88 na stránce 639 jsou použitelné pro JMS i pro XMS .NET.

Tabulka 88. Vlastnosti produktu XMS .NET

Vlastnost	Typ objektu				
	CF	**QCF-počet**	**TCF-zařízení**	**Fronta**	**Téma**
APPLICATIONNAME	Y	Y	Y	Není k dispozici	Není k dispozici
ASYNCEXCEPTION	Y	Y	Y	Není k dispozici	Není k dispozici
CCDTURL	Y	Y	Y	Není k dispozici	Není k dispozici
CHANNEL	Y	Y	Y	Není k dispozici	Není k dispozici
ConnectionNameList	Y	Y	Y	Není k dispozici	Není k dispozici
CLIENTRECONNECTOPTIONS	Y	Y	Y	Není k dispozici	Není k dispozici
CLIENTRECONNECTTIMEOUT	Y	Y	Y	Není k dispozici	Není k dispozici
CLIENTID	Není k dispozici	Y	Není k dispozici	Není k dispozici	Není k dispozici
COMPHDR "1" na stránce 640	Y	Není k dispozici	Y	Není k dispozici	Není k dispozici
COMPMSG "1" na stránce 640	Y	Y	Y	Není k dispozici	Není k dispozici
CONNOPT "1" na stránce 640	Y	Y	Y	Není k dispozici	Není k dispozici
CONNTAG "1" na stránce 640	Y	Y	Y	Není k dispozici	Není k dispozici
Popis"1" na stránce 640	Není k dispozici	Y	Není k dispozici	Y	Y
VYPRŠENÍ "1" na stránce 640	Není k dispozici	Není k dispozici	Není k dispozici	Y	Y
FAILIFQUIESCE	Y	Y	Y	Y	Y
název_hostitele	Není k dispozici	Y	Není k dispozici	Není k dispozici	Není k dispozici
LOCALADDRESS	Není k dispozici	Y	Není k dispozici	Není k dispozici	Není k dispozici
PERSISTENCE	Není k dispozici	Není k dispozici	Není k dispozici	Y	Y
PORT	Není k dispozici	Y	Není k dispozici	Není k dispozici	Není k dispozici
Priorita "1" na stránce 640	Není k dispozici	Není k dispozici	Není k dispozici	Y	Y
PROVIDERVERSION "1" na stránce 640	Není k dispozici	Y	Není k dispozici	Není k dispozici	Není k dispozici
QMANAGER	Y	Y	Y	Y	Není k dispozici
FRONTA "1" na stránce 640	Není k dispozici	Není k dispozici	Není k dispozici	Y	Není k dispozici

Tabulka 88. Vlastnosti produktu XMS .NET (pokračování)					
Vlastnost	**Typ objektu**				
	CF	**QCF-počet**	**TCF-zařízení**	**Fronta**	**Téma**
SHARECONVALLOWED	Y	Y	Y	Není k dispozici	Není k dispozici
Téma"1" na stránce 640	Není k dispozici	Není k dispozici	Není k dispozici	Není k dispozici	Y
Přenos"1" na stránce 640	Není k dispozici	Y	Není k dispozici	Není k dispozici	Není k dispozici

Poznámka:

1. Tyto vlastnosti nemají vlastnosti na úrovni aplikace, ale lze je volitelně nastavit pomocí spravovaných vlastností.

OutboundSNI vlastnost

V 9.3.0

V produktu IBM MQ 9.3.0můžete nastavit vlastnost XMSC_WMQ_OUTBOUND_SNI, která nastaví vlastnost **OutboundSNI** v aplikaci.

Vlastnost XMSC_WMQ_OUTBOUND_SNI_PROPERTY má následující hodnoty:

- XMSC_WMQ_OUTBOUND_SNI_CHANNEL, která se mapuje na "CHANNEL"
- XMSC_WMQ_OUTBOUND_SNI_HOSTNAME, který se mapuje na "HOSTNAME"
- XMSC_WMQ_OUTBOUND_SNI_ASTERISK, který mapuje na "*"

Dále můžete nastavit vlastnost **OutboundSNI** pomocí proměnné prostředí MQOUTBOUND_SNI, která má následující hodnoty:

- CHANNEL
- HOSTNAME
- *

Poznámka: Vlastnost je standardně nastavena na hodnotu XMSC_WMQ_OUTBOUND_SNI_CHANNEL, pokud není nastavena žádná specifická hodnota.

Pořadí priorit pro nastavení vlastnosti **OutboundSNI** ve spravovaném uzlu je:

1. Vlastnost na úrovni aplikace
2. Proměnná prostředí

Pro vlastnost **OutboundSNI** v nespravovaném uzlu je podporována pouze vlastnost `mqclient.ini` .

XMS .NET povinné vlastnosti pro spravované objekty ConnectionFactory

Když aplikace vytváří továrnu připojení, musí být definována řada vlastností, aby bylo možné vytvořit připojení k serveru systému zpráv.

Vlastnosti uvedené v následujících tabulkách jsou minimem nezbytným pro nastavení aplikace pro vytvoření připojení k serveru systému zpráv. Chcete-li upravit způsob vytvoření připojení, může vaše aplikace podle potřeby nastavit další vlastnosti objektu ConnectionFactory . Další informace viz Vlastnosti továrny připojení ConnectionFactory. Je uveden úplný seznam dostupných vlastností.

Připojení ke správci front IBM MQ

Tabulka 89. Nastavení vlastností pro spravované objekty ConnectionFactory pro připojení ke správci front IBM MQ	
Povinná vlastnost XMS	**Je vyžadována ekvivalentní vlastnost IBM MQ JMS .**
XMSC_CONNECTION_TYPE	XMS to vypracuje z názvu třídy továrny připojení a vlastnosti TRANSPORT (TRAN).
XMSC_WMQ_HOST_NAME	NÁZEV HOSTITELE (HOSTITEL)
XMSC_WMQ_PORT	PORT
XMSC_WMQ_QUEUE_MANAGER	Název správce front

Připojení ke zprostředkovateli v reálném čase

Tabulka 90. Nastavení vlastností pro spravované objekty ConnectionFactory pro připojení zprostředkovatele v reálném čase	
PovinnéXMS	**Je vyžadována ekvivalentní vlastnost IBM MQ JMS .**
XMSC_CONNECTION_TYPE	XMS to vypracuje z názvu třídy továrny připojení a vlastnosti TRANSPORT (TRAN).
XMSC_RTT_HOST_NAME	NÁZEV HOSTITELE (HOSTITEL)
XMSC_RTT_PORT	PORT

Připojení k serveru WebSphere Application Server service integration bus

Tabulka 91. Nastavení vlastností pro spravované objekty ConnectionFactory pro připojení k serveru WebSphere Application Server service integration bus	
XMS vlastnost	**Popis**
XMSC_CONNECTION_TYPE	Typ serveru systému zpráv, ke kterému se aplikace připojuje.. Tato hodnota je určena z názvu třídy továrny připojení.
XMSC_WPM_BUS_NAME	U továrny připojení se jedná o název sběrnice SIBus, ke které se aplikace připojuje, nebo pro cíl, název sběrnice SIBus, ve které cíl existuje.

XMS .NET požadované vlastnosti pro spravované cílové objekty

Aplikace, která vytváří cíl, musí nastavit několik vlastností, které aplikace ve spravovaném cílovém objektu.

Tabulka 92. Nastavení vlastností pro spravované cílové objekty		
Typ připojení	**Vlastnost**	**Popis**
IBM MQ správce front	QUEUE (QU) (fronta) TOPIC (TOP)	Fronta, ke které se chcete připojit Téma, které aplikace používá jako místo určení
Připojení ke zprostředkovateli v reálném čase	TOPIC (TOP)	Téma, které aplikace používá jako místo určení

Tabulka 92. Nastavení vlastností pro spravované cílové objekty (pokračování)		
Typ připojení	**Vlastnost**	**Popis**
WebSphere Application Server service integration bus	topicName queueName	Pokud se vaše aplikace připojuje k tématu Pokud se vaše aplikace připojuje k frontě

XMS .NET vytváření spravovaných objektů

Definice objektů ConnectionFactory a Destination, které aplikace XMS vyžadují pro vytvoření připojení k serveru systému zpráv, musí být vytvořeny pomocí příslušných administrativních nástrojů.

Než začnete

Další podrobnosti o různých typech spravovaného úložiště objektů, které produkt XMS podporuje, viz "XMS .NET podporované typy úložiště spravovaných objektů" na stránce 638.

Informace o této úloze

Chcete-li vytvořit spravované objekty pro produkt IBM MQ , použijte nástroj administrace IBM MQ Explorer nebo IBM MQ JMS (JMSAdmin).

Chcete-li vytvořit spravované objekty pro IBM MQ nebo IBM Integration Bus, použijte nástroj administrace platformy JMS IBM MQ (JMSAdmin).

Chcete-li vytvořit spravované objekty pro produkt WebSphere Application Server service integration bus, použijte administrativní konzolu WebSphere Application Server .

V administrativních nástrojích je vlastnost zkráceně označována jako **APPLICATIONNAME** nebo **APPNAME** .

Poznámka: K nastavení TRANSPORT (UNMANAGED) nelze použít JMSAdmin. Chcete-li tedy získat nespravovaného klienta XMS pomocí administrativně zvoleného názvu aplikace, musíte zadat následující příkaz:

```
cf.SetIntProperty(XMSC.WMQ_CONNECTION_MODE, XMSC.WMQ_CM_CLIENT_UNMANAGED);
```

Následující kroky shrnují, co děláte pro vytvoření spravovaných objektů.

Postup

1. Vytvořte továrnu připojení a definujte nezbytné vlastnosti pro vytvoření připojení z aplikace k vybranému serveru.

   Minimální vlastnosti, které produkt XMS vyžaduje k vytvoření připojení, jsou definovány v souboru "XMS .NET povinné vlastnosti pro spravované objekty ConnectionFactory" na stránce 640.
2. Vytvořte požadované místo určení na serveru systému zpráv, ke kterému se vaše aplikace připojuje:
   - Pro připojení ke správci front IBM MQ vytvořte frontu nebo téma.
   - Pro připojení v reálném čase ke zprostředkovateli vytvořte téma.
   - Pro připojení k serveru WebSphere Application Server service integration busvytvořte frontu nebo téma.

   Minimální vlastnosti, které produkt XMS vyžaduje k vytvoření připojení, jsou definovány v souboru "XMS .NET požadované vlastnosti pro spravované cílové objekty" na stránce 641.

XMS .NET vytváření objektů InitialContext

Aplikace musí vytvořit počáteční kontext, který se má použít k vytvoření připojení k úložišti spravovaných objektů pro načtení požadovaných spravovaných objektů.

Informace o této úloze

Objekt InitialContext zapouzdřuje připojení k úložišti. Rozhraní API XMS poskytuje metody k provedení následujících úloh:

- Vytvořit objekt InitialContext
- Vyhledejte spravovaný objekt v úložišti spravovaných objektů.

Procedura

- Další podrobnosti o vytvoření objektu InitialContext viz InitialContext pro .NET a Vlastnosti InitialContext.

Vlastnosti XMS .NET InitialContext

Parametry konstruktoru InitialContext zahrnují umístění úložiště spravovaných objektů dané jako identifikátor URI (Uniform Resource Indicator). Aby mohla aplikace navázat připojení k úložišti, může být nezbytné poskytnout více informací než informace obsažené v identifikátoru URI.

V rozhraní JNDI a v .NET implementaci XMSjsou konstruktoru poskytnuty další informace v hašovací tabulce prostředí.

Umístění spravovaného úložiště objektů je definováno ve vlastnosti XMSC_IC_URL . Tato vlastnost se obvykle předává při volání Create, ale lze ji upravit tak, aby se před vyhledáním připojila k jinému adresáři pojmenování. V případě kontextů FileSystem nebo LDAP tato vlastnost definuje adresu adresáře. V případě pojmenování COS se jedná o adresu webové služby, která používá tyto vlastnosti pro připojení k adresáři rozhraní JNDI.

Následující vlastnosti jsou předány nezměněné webové službě, která je použije pro připojení k adresáři rozhraní JNDI.

- XMSC_IC_PROVIDER_URL
- XMSC_IC_SECURITY_CREDENTIALS
- XMSC_IC_SECURITY_AUTHENTICATION
- XMSC_IC_SECURITY_PRINCIPAL
- XMSC_IC_SECURITY_PROTOCOL

Formát identifikátoru URI pro počáteční kontexty XMS

Umístění úložiště spravovaných objektů je poskytnuto jako identifikátor URI (Uniform Resource Indicator). Formát identifikátoru URI závisí na typu kontextu.

Kontext FileSystem

Pro kontext FileSystem URL uvádí umístění adresáře založeného na systému souborů. Struktura URL je definována v dokumentu RFC 1738, *Uniform Resource Locators (URL)*: URL má předponu `file://`a syntaxe následující za touto předponou je platnou definicí souboru, který lze otevřít v systému, na kterém je spuštěn produkt XMS .

Tato syntaxe může být specifická pro platformu a může používat oddělovače '/separators nebo' \ '. Použijete-li znak '\', musí být každý oddělovač změněn pomocí dalšího znaku '\'. To brání tomu, aby se rámec .NET pokusil interpretovat oddělovač jako řídicí znak pro to, co následuje.

Tyto příklady ilustrují tuto syntaxi:

```
file://myBindings
file:///admin/.bindings
file://\\admin\\.bindings
file://c:/admin/.bindings
file://c:\\admin\\.bindings
```

```
file://\\\\madison\\shared\\admin\\.bindings
file:///usr/admin/.bindings
```

Kontext LDAP

Pro kontext LDAP je základní struktura URL definována podle RFC 2255, *The LDAP URL Format*, s předponou bez rozlišování malých a velkých písmen `ldap://`

Přesná syntaxe je znázorněna v následujícím příkladu:

```
LDAP://[Hostname][:Port]["/"[DistinguishedName]]
```

Tato syntaxe je definována v RFC, ale bez podpory atributů, rozsahu, filtrů nebo rozšíření.

Příklady této syntaxe:

```
ldap://madison:389/cn=JMSData,dc=IBM,dc=UK
ldap://madison/cn=JMSData,dc=IBM,dc=UK
LDAP:///cn=JMSData,dc=IBM,dc=UK
```

Kontext služby WSS

Pro kontext WSS je URL ve formě koncového bodu webových služeb s předponou `http://`.

Případně můžete použít předponu `cosnaming://` nebo `wsvc://`,

Tyto dvě předpony jsou interpretovány tak, že používáte kontext WSS s URL , ke které se přistupuje přes http, což umožňuje snadné odvozování počátečního typu kontextu přímo z URL.

Příklady této syntaxe jsou následující:

```
http://madison.ibm.com:9080/xmsjndi/services/JndiLookup
cosnaming://madison/jndilookup
```

Webová služba vyhledání v rozhraní JNDI pro XMS .NET

Chcete-li přistupovat k adresáři pojmenování COS z adresáře XMS, musí být na serveru WebSphere Application Server service integration bus implementována webová služba JNDI Lookup. Tato webová služba převádí informace o produktu Java ze služby názvů COS do formátu, který mohou číst aplikace XMS .

Webová služba je poskytována v souboru podnikového archivu SIBXJndiLookupEAR.earumístěném v instalačním adresáři. Pro aktuální verzi souboru IBM MQ Message Service Client (XMS) for .NETlze soubor SIBXJndiLookupEAR.ear nalézt v adresáři *install_dir*`\java\lib` . Tuto instalaci lze provést na serveru WebSphere Application Server service integration bus pomocí administrativní konzoly nebo skriptovacího nástroje wsaadmin. Další informace o implementaci aplikací webových služeb naleznete v dokumentaci k produktu.

Chcete-li definovat webovou službu v aplikacích XMS , stačí nastavit vlastnost XMSC_IC_URL objektu InitialContext na adresu URL URL. Pokud je například webová služba implementována na hostiteli serveru s názvem MyServer, příklad koncového bodu webové služby URL:

```
wsvc://MyHost:9080/SIBXJndiLookup/services/JndiLookup
```

Nastavení vlastnosti XMSC_IC_URL umožňuje voláním funkce InitialContext Lookup vyvolat webovou službu na definovaném koncovém bodu, což následně vyhledá požadovaný spravovaný objekt ze služby názvů COS.

Aplikace .NET mohou používat webovou službu. Implementace na straně serveru je stejná pro XMS C, /C++ a XMS .NET. XMS Produkt .NET vyvolává webovou službu přímo prostřednictvím konzoly Microsoft .NET Framework.

XMS .NET načtení spravovaných objektů

Produkt XMS načte spravovaný objekt z úložiště s použitím adresy poskytnuté při vytvoření objektu InitialContext nebo ve vlastnostech InitialContext .

Objekty, které mají být načteny, mohou mít následující typy názvů:

- Jednoduchý název popisující cílový objekt, například cíl fronty s názvem SalesOrders .
- Složený název, který může být tvořen SubContexts, oddělený znakem '/', a musí končit názvem objektu. Příkladem složeného názvu je "Warehouse/PickLists/DispatchQueue2", kde Warehouse a Picklists jsou SubContexts v adresáři pojmenování a DispatchQueue2 je název cílového objektu.

Zabránění aplikacím v použití novější verze produktu XMS

Standardně, když je instalována novější verze produktu XMS , aplikace používající předchozí verzi se automaticky přepnou na novější verzi, aniž by bylo nutné recompile.Hovevervšak můžete aplikacím zabránit v použití novější verze nastavením atributu v konfiguračním souboru aplikace.

Informace o této úloze

Funkce koexistence více verzí zajišťuje, že instalace novější verze produktu XMS nepřepíše předchozí verzi produktu XMS . Místo toho existuje více instancí podobných sestavení XMS .NET současně v globální mezipaměti sestavení (GAC), ale mají různá čísla verzí. Interně používá GAC soubor zásad ke směrování volání aplikace na nejnovější verzi produktu XMS. Aplikace se spouštějí bez nutnosti rekompilace a mohou používat nové funkce dostupné v novější verzi produktu XMS .NET .

Procedura

- Pokud se požaduje, aby aplikace používala starší verzi produktu XMS .NET , nastavte atribut `publisherpolicy` v konfiguračním souboru aplikace na hodnotu no .

  Poznámka: Konfigurační soubor aplikace je soubor s názvem, který se skládá z názvu spustitelného programu, ke kterému se soubor vztahuje, s příponou .config. Například konfigurační soubor aplikace text.exe by měl název text.exe.config.

  Kdykoli však všechny aplikace systému používají stejnou verzi produktu XMS .NET.

Zabezpečení komunikací pro aplikace XMS

V tomto oddílu jsou uvedeny informace o nastavení zabezpečených komunikací, které umožní aplikacím XMS připojit se prostřednictvím zabezpečení SSL (Secure Sockets Layer) ke stroji systému zpráv WebSphere Application Server service integration bus nebo ke správci front IBM MQ .

Informace o této úloze

Sekce obsahuje následující témata:

- “Zabezpečená připojení ke správci front IBM MQ” na stránce 646
- “Mapování názvů CipherSuite a CipherSpec pro připojení produktu XMS ke správci front IBM MQ” na stránce 646
- “Zabezpečená připojení ke stroji systému zpráv WebSphere Application Server service integration bus” na stránce 647
- “Mapování názvů CipherSuite a CipherSpec pro připojení k serveru WebSphere Application Server service integration bus” na stránce 648

Zabezpečená připojení ke správci front IBM MQ

Chcete-li povolit aplikaci XMS .NET vytvářet zabezpečená připojení ke správci front IBM MQ , příslušné vlastnosti musí být definovány v objektu ConnectionFactory .

Protokol použitý při vyjednávání šifrování může být buď SSL (Secure Sockets Layer), nebo TLS (Transport Layer Security), v závislosti na tom, kterou sadu CipherSuite zadáte v objektu ConnectionFactory .

Vlastnosti ConnectionFactory pro připojení pomocí SSL ke správci front IBM MQ se stručným popisem jsou uvedeny v následující tabulce:

Tabulka 93. Vlastnosti ConnectionFactory pro připojení ke správci front IBM MQ prostřednictvím zabezpečení SSL

Název majetku	Popis
XMSC_WMQ_SSL_CERT_STORES	Umístění serverů obsahující seznamy odvolaných certifikátů (CRL), které mají být použity v připojení SSL ke správci front.
XMSC_WMQ_SSL_CIPHER_SPEC	Název specifikace CipherSpec, která má být použita v zabezpečeném připojení ke správci front.
XMSC_WMQ_SSL_CIPHER_SUITE	Název sady CipherSuite, která má být použita v rámci připojení TLS ke správci front. Protokol použitý při vyjednávání zabezpečeného připojení závisí na určené sadě CipherSuite.
XMSC_WMQ_SSL_CRYPTO_HW	Podrobnosti konfigurace šifrovacího hardwaru připojeného k systému klienta.
XMSC_WMQ_SSL_FIPS_REQUIRED	Hodnota této vlastnosti určuje, zda aplikace může nebo nemůže používat šifrovací sady, které nevyhovují standardu FIPS. Je-li tato vlastnost nastavena na hodnotu true, pro připojení klientského serveru se používají pouze algoritmy FIPS.
XMSC_WMQ_SSL_KEY_REPOSITORY	Umístění souboru databáze klíčů, ve kterém jsou uloženy klíče a certifikáty.
XMSC_WMQ_SSL_KEY_RESETCOUNT	Hodnota KeyResetCount představuje celkový počet nešifrovaných bajtů odeslaných a přijatých v rámci konverzace SSL, než je znovu vyjednán tajný klíč.
XMSC_WMQ_SSL_PEER_NAME	Název typu peer, které se použije při připojení SSL ke správci front.

Mapování názvů CipherSuite a CipherSpec pro připojení produktu XMS ke správci front IBM MQ

InitialContext se překládá mezi vlastností továrny připojení JMSAdmin SSLCIPHERSUITE a téměř ekvivalentní hodnotou XMS XMSC_WMQ_SSL_CIPHER_SPEC. Podobný překlad je nezbytný, pokud zadáte hodnotu pro XMSC_WMQ_SSL_CIPHER_SUITE, ale vynecháte hodnotu pro XMSC_WMQ_SSL_CIPHER_SPEC.

Tabulka 94 na stránce 646 vypisuje dostupné CipherSpecs a jejich ekvivalenty JSSE CipherSuite .

Tabulka 94. Dostupné CipherSpecs a jejich ekvivalenty JSSE CipherSuite

CipherSpec	Ekvivalentní sada JSSE CipherSuite
TLS_RSA_WITH_3DES_EDE_CBC_SHA	SSL_RSA_WITH_3DES_EDE_CBC_SHA
TLS_RSA_WITH_AES_128_CBC_SHA	SSL_RSA_WITH_AES_128_CBC_SHA

Tabulka 94. Dostupné CipherSpecs a jejich ekvivalenty JSSE CipherSuite (pokračování)	
CipherSpec	**Ekvivalentní sada JSSE CipherSuite**
TLS_RSA_WITH_AES_256_CBC_SHA	SSL_RSA_WITH_AES_256_CBC_SHA
TLS_RSA_WITH_DES_CBC_SHA	SSL_RSA_WITH_DES_CBC_SHA

Poznámka: Deprecated TLS_RSA_WITH_3DES_EDE_CBC_SHA je zamítnutý. Přesto jej lze použít k přenosu až 32 GB dat před ukončením připojení s chybou AMQ9288. Chcete-li se vyhnout této chybě, musíte se buď vyhnout použití trojitého DES, nebo povolit reset tajného klíče při použití této CipherSpec.

Zabezpečená připojení ke stroji systému zpráv WebSphere Application Server service integration bus

Chcete-li povolit aplikaci XMS .NET vytvářet zabezpečená připojení ke stroji systému zpráv WebSphere Application Server service integration bus , musí být příslušné vlastnosti definovány v objektu ConnectionFactory .

Produkt XMS poskytuje podporu SSL a HTTPS pro připojení k serveru WebSphere Application Server service integration bus. SSL a HTTPS poskytují zabezpečená připojení pro ověření a utajení.

Podobně jako zabezpečení WebSphere je zabezpečení XMS konfigurováno s ohledem na standardy zabezpečení JSSE a konvence pojmenování, které zahrnují použití CipherSuites k určení algoritmů, které se používají při vyjednávání zabezpečeného připojení. Protokol použitý při vyjednávání šifrování může být buď SSL, nebo TLS, v závislosti na tom, kterou sadu CipherSuite zadáte v objektu ConnectionFactory .

Tabulka 95 na stránce 647 uvádí vlastnosti, které musí být definovány v objektu ConnectionFactory .

Tabulka 95. Vlastnosti ConnectionFactory pro zabezpečená připojení ke stroji systému zpráv WebSphere Application Server service integration bus	
Název majetku	**Popis**
XMSC_WPM_SSL_CIPHER_SUITE	Název sady CipherSuite , která má být použita pro připojení TLS ke stroji systému zpráv WebSphere Application Server service integration bus . Protokol použitý při vyjednávání zabezpečeného připojení závisí na určené sadě CipherSuite.
XMSC_WPM_SSL_KEYRING_LABEL	Certifikát, který má být použit při ověřování na serveru.

Následuje příklad vlastností ConnectionFactory pro zabezpečená připojení ke stroji systému zpráv WebSphere Application Server service integration bus :

```
cf.setStringProperty(XMSC_WPM_PROVIDER_ENDPOINTS, host_name:port_number:chain_name);
cf.setStringProperty(XMSC_WPM_SSL_KEY_REPOSITORY, key_repository_pathname);
cf.setStringProperty(XMSC_WPM_TARGET_TRANSPORT_CHAIN, transport_chain);
cf.setStringProperty(XMSC_WPM_SSL_CIPHER_SUITE, cipher_suite);
cf.setStringProperty(XMSC_WPM_SSL_KEYRING_STASH_FILE, stash_file_pathname);
```

Kde název řetězu by měl být nastaven na BootstrapTunneledSecureMessaging nebo BootstrapSecureMessaging a číslo_portu je číslo portu, na kterém server samozavedení naslouchá příchozím požadavkům.

Následuje příklad vlastností ConnectionFactory pro zabezpečená připojení ke stroji systému zpráv WebSphere Application Server service integration bus s vloženými ukázkovými hodnotami:

```
  /* CF properties needed for an SSL connection */
    cf.setStringProperty(XMSC_WPM_PROVIDER_ENDPOINTS,"localhost:7286:BootstrapSecureMessaging");
    cf.setStringProperty(XMSC_WPM_TARGET_TRANSPORT_CHAIN,"InboundSecureMessaging");
    cf.setStringProperty(XMSC_WPM_SSL_KEY_REPOSITORY,"C:\\Program Files\\IBM\\gsk7\\bin\
\XMSkey.kdb");
    cf.setStringProperty(XMSC_WPM_SSL_KEYRING_STASH_FILE,"C:\\Program Files\\IBM\\gsk7\\bin\
```

```
\XMSkey.sth");
    cf.setStringProperty(XMSC_WPM_SSL_CIPHER_SUITE,"SSL_RSA_EXPORT_WITH_RC4_40_MD5");
```

Mapování názvů CipherSuite a CipherSpec pro připojení k serveru WebSphere Application Server service integration bus

Vzhledem k tomu, že produkt IBM Global Security Kit (GSKit) používá spíše CipherSpecs než CipherSuites, musí být názvy CipherSuite ve stylu JSSE určené ve vlastnosti XMSC_WPM_SSL_CIPHER_SUITE mapovány na názvy GSKit-style CipherSpec .

Tabulka 96 na stránce 648 uvádí ekvivalentní CipherSpec pro každou rozpoznanou CipherSuite.

Tabulka 96. Dostupné CipherSuites a jejich ekvivalentní CipherSpecs

CipherSuite	**CipherSpec ekvivalentní**
TLS_RSA_WITH_DES_CBC_SHA	TLS_RSA_WITH_DES_CBC_SHA
TLS_RSA_WITH_3DES_EDE_CBC_SHA	TLS_RSA_WITH_3DES_EDE_CBC_SHA
TLS_RSA_WITH_AES_128_CBC_SHA	TLS_RSA_WITH_AES_128_CBC_SHA
TLS_RSA_WITH_AES_256_CBC_SHA	TLS_RSA_WITH_AES_256_CBC_SHA

Poznámka: Deprecated TLS_RSA_WITH_3DES_EDE_CBC_SHA je zamítnutý. Přesto jej lze použít k přenosu až 32 GB dat před ukončením připojení s chybou AMQ9288. Chcete-li se vyhnout této chybě, musíte se vyhnout použití trojitého DES nebo povolit reset tajného klíče při použití této CipherSpec.

Zprávy produktu XMS

Tento oddíl popisuje strukturu a obsah zpráv XMS a vysvětluje, jak aplikace zpracovávají zprávy XMS .

Tento oddíl obsahuje následující témata:

- “Části zprávy XMS” na stránce 648
- “Pole záhlaví ve zprávě XMS” na stránce 649
- “Vlastnosti zprávy XMS” na stránce 649
- “Tělo zprávy XMS” na stránce 653
- “Selektory zpráv” na stránce 655
- “Mapování zpráv XMS na zprávy IBM MQ” na stránce 656

Části zprávy XMS

Zpráva XMS se skládá ze záhlaví, sady vlastností a těla.

Header
Záhlaví zprávy obsahuje pole a všechny zprávy obsahují stejnou sadu polí záhlaví. Produkt XMS a aplikace používají hodnoty polí záhlaví k identifikaci a směrování zpráv. Další informace o polích záhlaví viz “Pole záhlaví ve zprávě XMS” na stránce 649.

Sada vlastností
Vlastnosti zprávy uvádějí další informace o zprávě. Ačkoli všechny zprávy mají stejnou sadu polí záhlaví, každá zpráva může mít jinou sadu vlastností. Další informace viz “Vlastnosti zprávy XMS” na stránce 649.

Tělo
Tělo zprávy obsahuje data aplikace. Další informace viz “Tělo zprávy XMS” na stránce 653.

Aplikace může vybrat, které zprávy chce přijmout. Pomocí selektorů zpráv, které určují kritéria výběru. Kritéria mohou být založena na hodnotách určitých polí záhlaví a na hodnotách všech vlastností zprávy. Další informace o selektorech zpráv viz “Selektory zpráv” na stránce 655.

Pole záhlaví ve zprávě XMS

Chcete-li povolit aplikaci XMS výměnu zpráv s aplikací WebSphere JMS , záhlaví zprávy XMS obsahuje pole záhlaví zprávy JMS .

Názvy těchto polí záhlaví začínají předponou JMS. Popis polí záhlaví zprávy JMS viz *Java Message Service Specifikace*.

XMS implementuje pole záhlaví zprávy JMS jako atributy objektu zprávy. Každé pole záhlaví má své vlastní metody pro nastavení a získání své hodnoty. Popis těchto metod viz IMessage. Pole záhlaví je vždy čitelné a zapisovatelné.

Tabulka 97 na stránce 649 vypisuje JMS pole záhlaví zprávy a označuje, jak je hodnota každého pole nastavena pro přenesenou zprávu. Některá pole jsou automaticky nastavena produktem XMS , když aplikace odešle zprávu, nebo, v případě JMSRedelivered, když aplikace obdrží zprávu.

Tabulka 97. JMS pole záhlaví zprávy.]

Název pole záhlaví zprávy JMS	**Jak je hodnota nastavena pro přenesenou zprávu (ve formátu *metoda* [*class*])**
JMSCorrelationID	Nastavit JMSCorrelationID [Zpráva]
JMSDeliveryMode	Odeslat [MessageProducer]
JMSDestination	Odeslat [MessageProducer]
JMSExpiration	Odeslat [MessageProducer]
JMSMessageID	Odeslat [MessageProducer]
JMSPriority.	Odeslat [MessageProducer]
JMSRedelivered	Přijmout [MessageConsumer]
JMSReplyTo	Nastavit JMSReplyTo [Zpráva]
JMSTimestamp	Odeslat [MessageProducer]
JMSType.	Nastavit JMSType [Zpráva]

Vlastnosti zprávy XMS

Produkt XMS podporuje tři druhy vlastností zpráv: JMS definované vlastnosti, IBM definované vlastnosti a vlastnosti definované aplikací.

Aplikace XMS může vyměňovat zprávy s aplikací WebSphere JMS , protože produkt XMS podporuje následující předdefinované vlastnosti objektu zprávy:

- Stejné JMS-definované vlastnosti, které produkt WebSphere JMS podporuje. Názvy těchto vlastností začínají předponou JMSX.
- Stejné IBM-definované vlastnosti, které produkt WebSphere JMS podporuje. Názvy těchto vlastností začínají předponou JMS_IBM_.

Každá předdefinovaná vlastnost má dva názvy:

- Název JMS pro vlastnost JMS-defined nebo název WebSphere JMS pro vlastnost IBM-defined.

  Jedná se o název, pod kterým je vlastnost známa v produktu JMS nebo WebSphere JMS, a jedná se také o název, který je přenášen se zprávou, která tuto vlastnost obsahuje. Aplikace XMS používá tento název k identifikaci vlastnosti ve výrazu selektoru zpráv.
- Název XMS pro identifikaci vlastnosti ve všech situacích kromě výrazu selektoru zpráv. Každý název XMS je definován jako pojmenovaná konstanta ve třídě `IBM.XMS.XMSC` . Hodnota pojmenované konstanty je odpovídající JMS nebo WebSphere JMS název.

Kromě předdefinovaných vlastností může aplikace XMS vytvářet a používat vlastní sadu vlastností zpráv. Tyto vlastnosti se nazývají *vlastnosti definované aplikací.*

Po vytvoření zprávy aplikací jsou vlastnosti zprávy čitelné a schopné zápisu. Vlastnosti zůstávají čitelné a lze do nich zapisovat i po odeslání zprávy aplikací. Když aplikace přijme zprávu, vlastnosti zprávy jsou jen pro čtení. Pokud aplikace volá metodu `Clear Properties` třídy Message, když jsou vlastnosti zprávy jen pro čtení, budou tyto vlastnosti čitelné a schopné zápisu. Tato metoda také vymaže vlastnosti.

Přijatá zpráva po postoupení po vymazání vlastností zprávy se bude chovat způsobem, který je konzistentní s chováním předávání standardního WMQ XMS pro .NET BytesMessage s vymazanými vlastnostmi zprávy.

To se však nedoporučuje, protože následující vlastnosti budou ztraceny:

- Hodnota vlastnosti JMS_IBM_Encoding znamená, že data zprávy nelze smysluplně dekódovat.
- Hodnota vlastnosti JMS_IBM_Format, která znamená, že řetězení záhlaví mezi záhlavím zprávy (MQMD nebo novým MQRFH2) a existujícími záhlavími bude přerušeno.

Chcete-li určit hodnoty všech vlastností zprávy, může aplikace volat metodu Get Properties třídy Message. Tato metoda vytvoří iterátor, který zapouzdří seznam objektů Vlastnost, kde každý objekt Vlastnost představuje vlastnost zprávy. Aplikace pak může pomocí metod třídy iterátoru postupně načíst každý objekt vlastnosti a může použít metody třídy vlastnosti k načtení názvu, datového typu a hodnoty každé vlastnosti.

JMS-definované vlastnosti zprávy

Několik vlastností zprávy definovaných službou JMS je podporováno produktem XMS i produktem WebSphere JMS.

Tabulka 98 na stránce 650 obsahuje seznam vlastností zprávy definovaných službou JMS, které jsou podporovány produktem XMS i produktem WebSphere JMS. Popis JMS-definovaných vlastností viz *Java Message Service Specifikace*. Vlastnosti definované parametrem JMSnejsou platné pro připojení v reálném čase ke zprostředkovateli.

Tabulka určuje datový typ každé vlastnosti a určuje, jak je hodnota vlastnosti nastavena pro přenášenou zprávu. Některé vlastnosti jsou nastaveny automaticky produktem XMS , když aplikace odešle zprávu, nebo, v případě JMSXDeliveryCount, když aplikace obdrží zprávu.

Tabulka 98. JMS-definované vlastnosti zprávy

XMS název definované vlastnosti JMS	**Název JMS**	**Datový typ**	**Jak je hodnota nastavena pro přenesenou zprávu (ve formátu *metoda* [*class*])**
JMSX_APPID	JMSXAppID	System.String	Odeslat [MessageProducer]
POČET_DORUČENÍ_ZPRÁV	JMSXDeliveryCount	System.Int32	Přijmout [MessageConsumer]
JMSX_GROUPID	JMSXGroupID	System.String	Nastavit vlastnost řetězce [PropertyContext]
JMSX_GROUPSEQ	JMSXGroupSeq	System.Int32	Nastavit celočíselnou vlastnost [PropertyContext]
JMSX_USERID	JMSXUserID	System.String	Odeslat [MessageProducer]

IBM-definované vlastnosti zprávy

XMS a WebSphere JMSpodporují některé vlastnosti zprávy definované společností IBM.

Tabulka 99 na stránce 651 vypíše IBM definované vlastnosti zprávy, které jsou podporovány produktem XMS i produktem WebSphere JMS. Další informace o vlastnostech definovaných v souboru IBMnaleznete v dokumentaci k produktu IBM MQ nebo WebSphere Application Server .

Tabulka určuje datový typ každé vlastnosti a určuje, jak je hodnota vlastnosti nastavena pro přenášenou zprávu. Některé vlastnosti jsou automaticky nastaveny produktem XMS , když aplikace odešle zprávu.

Tabulka 99. IBM-definované vlastnosti zprávy

XMS název IBM definované vlastnosti	**WebSphere JMS název**	**Datový typ**	**Jak je hodnota nastavena pro přenesenou zprávu (ve formátu *metoda* [*class*])**
JMS_IBM_CHARACTER_SET	JMS_IBM_Character_Set	System.Int32	Nastavit celočíselnou vlastnost [PropertyContext]
JMS_IBM_Encoding	JMS_IBM_Encoding	System.Int32	Nastavit celočíselnou vlastnost [PropertyContext]
Zpráva JMS_IBM_EXCEPTIONMESSAGE	JMS_IBM_ExceptionMessage	System.String	Přijmout [MessageConsumer]
JMS_IBM_EXCEPTIONREASON	JMS_IBM_ExceptionReason	System.Int32	Přijmout [MessageConsumer]
ČASOVÉ razítko JMS_IBM_EXCEPTIONTIMESTAMP	JMS_IBM_ExceptionTimestamp	System.Int64	Přijmout [MessageConsumer]
Problém s výjimkou JMS_IBM_EXCEPTIONPROBLEM DESTINATION	JMS_IBM_ExceptionProblemDestination	System.String	Přijmout [MessageConsumer]
ZPĚTNÁ vazba jms_ibm_feedback	Zpětná vazba JMS_IBM_Feedback	System.Int32	Nastavit celočíselnou vlastnost [PropertyContext]
JMS_IBM_FORMAT	Formát JMS_IBM_Format	System.String	Nastavit vlastnost řetězce [PropertyContext]
JMS_IBM_LAST_MSG_IN_GROUP	JMS_IBM_Last_Msg_In_Group	System.Boolean	Nastavit celočíselnou vlastnost [PropertyContext]
JMS_IBM_MSGTYPE	JMS_IBM_MsgType	System.Int32	Nastavit celočíselnou vlastnost [PropertyContext]
JMS_IBM_PUTAPPLTYPE	Typ JMS_IBM_PutAppl	System.Int32	Odeslat [MessageProducer]
JMS_IBM_PUTDATE	JMS_IBM_PutDate	System.String	Odeslat [MessageProducer]
JMS_IBM_PUTTIME	JMS_IBM_PutTime	System.String	Odeslat [MessageProducer]

Tabulka 99. IBM-definované vlastnosti zprávy (pokračování)

XMS název IBM definované vlastnosti	WebSphere JMS název	Datový typ	Jak je hodnota nastavena pro přenesenou zprávu (ve formátu *metoda* [*class*])
JMS_IBM_REPORT_COA	JMS_IBM_Report_COA	System.Int32	Nastavit celočíselnou vlastnost [PropertyContext]
JMS_IBM_REPORT_COD	JMS_IBM_Report_COD	System.Int32	Nastavit celočíselnou vlastnost [PropertyContext]
JMS_IBM_REPORT_DISCARD_MSG	JMS_IBM_Report_Discard_Msg	System.Int32	Nastavit celočíselnou vlastnost [PropertyContext]
VÝJIMKA-JMS_IBM_REPORT_EXCEPTION	Výjimka JMS_IBM_Report_Exception	System.Int32	Nastavit celočíselnou vlastnost [PropertyContext]
JMS_IBM_REPORT_EXPIRATION	JMS_IBM_Report_Expiration	System.Int32	Nastavit celočíselnou vlastnost [PropertyContext]
JMS_IBM_REPORT_NAN	JMS_IBM_Report_NAN	System.Int32	Nastavit celočíselnou vlastnost [PropertyContext]
JMS_IBM_REPORT_PAN	JMS_IBM_Report_PAN	System.Int32	Nastavit celočíselnou vlastnost [PropertyContext]
JMS_IBM_REPORT_PASS_CORREL_ ID	JMS_IBM_Report_Pass_Correl_ID	System.Int32	Nastavit celočíselnou vlastnost [PropertyContext]
JMS_IBM_REPORT_PASS_MSG_ID	JMS_IBM_Report_Pass_Msg_ID	System.Int32	Nastavit celočíselnou vlastnost [PropertyContext]
ID_SYSTÉMOVÉ_ZPRÁVY JMS_IBM_SYSTEM_MESSAGEID	JMS_IBM_System_MessageID	System.String	Odeslat [MessageProducer]

Vlastnosti zprávy definované aplikací

Aplikace XMS může vytvářet a používat vlastní sadu vlastností zpráv. Když aplikace odešle zprávu, tyto vlastnosti se také přenesou se zprávou. Přijímající aplikace, používající selektory zpráv, pak může vybrat, které zprávy chce přijmout, na základě hodnot těchto vlastností.

Chcete-li povolit aplikaci WebSphere JMS vybírat a zpracovávat zprávy odeslané aplikací XMS , musí název vlastnosti definované aplikací odpovídat pravidlům pro vytváření identifikátorů ve výrazech selektoru zpráv. Další informace viz “Selektory zpráv v adresáři JMS” na stránce 140. Hodnota vlastnosti definované aplikací musí mít jeden z následujících datových typů: System.Boolean, System.SByte, System.Int16, System.Int32, System.Int64, System.Float, System.Doublenebo System.String.

Tělo zprávy XMS

Tělo zprávy obsahuje data aplikace. Zpráva však nemůže obsahovat žádné tělo a může obsahovat pouze pole a vlastnosti záhlaví.

Produkt XMS podporuje pět typů těla zprávy:

Bajty
Tělo obsahuje proud bajtů. Zpráva s tímto typem těla se nazývá *bajtová zpráva*. Rozhraní IBytesMessage obsahuje metody pro zpracování těla bajtové zprávy.

Mapa
Tělo obsahuje sadu dvojic název-hodnota, kde každá hodnota má přidružený datový typ. Zpráva s tímto typem těla se nazývá *mapová zpráva*. Rozhraní IMapMessage obsahuje metody pro zpracování těla zprávy mapy.

Objekt
Tělo obsahuje serializovaný objekt Java nebo .NET . Zpráva s tímto typem těla se nazývá *zpráva objektu*. Rozhraní IObjectMessage obsahuje metody pro zpracování těla zprávy objektu.

Proud
Tělo obsahuje proud hodnot, kde každá hodnota má přidružený datový typ. Zpráva s tímto typem těla se nazývá *zpráva proudu*. Rozhraní IStreamMessage obsahuje metody pro zpracování těla zprávy proudu.

Text
Tělo obsahuje řetězec. Zpráva s tímto typem těla se nazývá *textová zpráva*. Rozhraní ITextMessage obsahuje metody pro zpracování těla textové zprávy.

Rozhraní IMessage je nadřízeným prvkem všech objektů zpráv a lze jej použít ve funkcích systému zpráv k reprezentaci libovolného typu zpráv XMS .

Informace o velikosti a maximální a minimální hodnotě každého z těchto datových typů viz Tabulka 84 na stránce 629.

Bajtové zprávy

Tělo bajtové zprávy obsahuje proud bajtů. Tělo obsahuje pouze skutečná data a je odpovědností odesílajících a přijímajících aplikací interpretovat tato data.

Bajtové zprávy jsou užitečné, pokud aplikace XMS potřebuje vyměňovat zprávy s aplikacemi, které nepoužívají rozhraní pro programování aplikací XMS nebo JMS .

Poté, co aplikace vytvoří bajtovou zprávu, tělo zprávy je pouze pro zápis. Aplikace sestaví data aplikace do těla voláním příslušných metod zápisu rozhraní IBytesMessage pro .NET. Pokaždé, když aplikace zapíše hodnotu do proudu bajtů zpráv, je hodnota sestavena okamžitě po předchozí hodnotě zapsané aplikací. Produkt XMS udržuje interní kurzor, aby si zapamatoval pozici posledního sestaveného bajtu.

Když aplikace odešle zprávu, tělo zprávy bude jen pro čtení. V tomto režimu může aplikace zprávu odesílat opakovaně.

Když aplikace obdrží bajtovou zprávu, tělo zprávy je jen pro čtení. Aplikace může použít odpovídající metody čtení rozhraní IBytesMessage ke čtení obsahu proudu bajtových zpráv. Aplikace čte bajty v posloupnosti a produkt XMS udržuje interní kurzor, aby si zapamatoval pozici posledního přečteného bajtu.

Pokud aplikace volá metodu Reset rozhraní IBytesMessage , když je tělo bajtové zprávy zapisovatelné, tělo bude jen pro čtení. Tato metoda také přemístí kurzor na začátek bajtových proudů zpráv.

Pokud aplikace volá metodu `Clear Body` rozhraní IMessage pro systém .NET , když je tělo bajtové zprávy jen pro čtení, tělo se stane zapisovatelným. Metoda také vymaže tělo.

Mapovat zprávy

Tělo zprávy mapy obsahuje sadu dvojic název-hodnota, kde každá hodnota má přidružený datový typ.

V každé dvojici název-hodnota je název řetězec, který identifikuje hodnotu, a hodnota je prvek dat aplikace, který má jeden z datových typů XMS uvedených v seznamu Tabulka 100 na stránce 655. Pořadí dvojic název-hodnota není definováno. Třída MapMessage obsahuje metody pro nastavení a získání dvojic název-hodnota.

Aplikace může přistupovat ke dvojici název-hodnota náhodně zadáním jejího názvu.

Aplikace .NET může pomocí vlastnosti MapNames získat výčet názvů v těle zprávy mapy.

Když aplikace získá hodnotu dvojice název-hodnota, může být hodnota převedena produktem XMS na jiný datový typ. Chcete-li například získat celé číslo z těla zprávy mapy, může aplikace volat metodu GetString třídy MapMessage , která vrací celé číslo jako řetězec. Podporované převody jsou stejné jako ty, které jsou podporovány, když produkt XMS převádí hodnotu vlastnosti z jednoho datového typu na jiný. Další informace o podporovaných převodech viz “Implicitní převod hodnoty vlastnosti z jednoho datového typu na jiný” na stránce 629.

Poté, co aplikace vytvoří zprávu mapy, tělo zprávy je čitelné a zapisovatelné. Tělo zůstává čitelné a zapisovatelné poté, co aplikace odešle zprávu. Když aplikace obdrží zprávu mapy, tělo zprávy je jen pro čtení. Pokud aplikace volá metodu Vymazat tělo třídy zprávy, když je tělo zprávy mapy jen pro čtení, tělo se stane čitelným a zapisovatelným. Metoda také vymaže tělo.

Zprávy objektů

Tělo zprávy objektu obsahuje serializovaný objektJava nebo .NET .

Aplikace XMS může přijmout zprávu objektu, změnit její pole záhlaví a vlastnosti a poté ji odeslat do jiného místa určení. Aplikace může také zkopírovat tělo zprávy objektu a použít jej k vytvoření jiné zprávy objektu. Produkt XMS považuje tělo zprávy objektu za pole bajtů.

Poté, co aplikace vytvoří zprávu objektu, je tělo zprávy čitelné a zapisovatelné. Tělo zůstává čitelné a zapisovatelné poté, co aplikace odešle zprávu. Když aplikace přijme zprávu objektu, tělo zprávy je jen pro čtení. Pokud aplikace volá metodu `Clear Body` rozhraní IMessage pro systém .NET , když je tělo zprávy objektu jen pro čtení, tělo bude čitelné a schopné zápisu. Metoda také vymaže tělo.

Proudové zprávy

Tělo zprávy proudu obsahuje proud hodnot, kde každá hodnota má přidružený datový typ.

Datový typ hodnoty je jeden z datových typů XMS uvedených v seznamu Tabulka 100 na stránce 655.

Poté, co aplikace vytvoří zprávu proudu, je tělo zprávy schopné zápisu. Aplikace sestavuje data aplikace do těla voláním příslušných metod zápisu rozhraní IStreamMessage pro .NET. Pokaždé, když aplikace zapíše hodnotu do proudu zpráv, hodnota a její datový typ se sestaví bezprostředně po předchozí hodnotě zapsané aplikací. Produkt XMS udržuje interní kurzor, aby si zapamatoval pozici poslední sestavené hodnoty.

Když aplikace odešle zprávu, tělo zprávy bude jen pro čtení. V tomto režimu může aplikace odeslat zprávu vícekrát.

Když aplikace přijme zprávu proudu, tělo zprávy je jen pro čtení. Aplikace může použít příslušné metody čtení rozhraní IStreamMessage pro produkt .NET ke čtení obsahu proudu zpráv. Aplikace čte hodnoty v pořadí a produkt XMS udržuje interní kurzor, aby si zapamatoval pozici poslední hodnoty, která byla přečtena.

Když aplikace přečte hodnotu z proudu zpráv, může ji produkt XMS převést na jiný datový typ. Chcete-li například načíst celé číslo z proudu zpráv, může aplikace volat metodu ReadString , která vrací celé číslo jako řetězec. Podporované převody jsou stejné jako ty, které jsou podporovány, když produkt XMS převádí hodnotu vlastnosti z jednoho datového typu na jiný. Další informace o podporovaných převodech viz “Implicitní převod hodnoty vlastnosti z jednoho datového typu na jiný” na stránce 629.

Pokud dojde k chybě při pokusu aplikace o čtení hodnoty z proudu zpráv, kurzor není rozšířený. Aplikace se může zotavit z chyby tím, že se pokusí přečíst hodnotu jako jiný datový typ.

Pokud aplikace volá metodu Reset rozhraní IStreamMessage pro systém XMS , když je tělo zprávy proudu jen pro zápis, tělo bude jen pro čtení. Tato metoda také přemístí kurzor na začátek proudu zpráv.

Pokud aplikace volá metodu `Clear Body` rozhraní IMessage pro XMS , když je tělo zprávy proudu jen pro čtení, tělo se stane jen pro zápis. Metoda také vymaže tělo.

Textové zprávy

Tělo textové zprávy obsahuje řetězec.

Poté, co aplikace vytvoří textovou zprávu, je tělo zprávy čitelné a lze do něj zapisovat. Tělo zůstává čitelné a zapisovatelné poté, co aplikace odešle zprávu. Když aplikace obdrží textovou zprávu, tělo zprávy je jen pro čtení. Pokud aplikace volá metodu Clear Body rozhraní IMessage pro .NET , když je tělo textové zprávy jen pro čtení, tělo se stane čitelným a zapisovatelným. Metoda také vymaže tělo.

Datové typy pro prvky dat aplikace

Chcete-li zajistit, aby si aplikace XMS mohla vyměňovat zprávy s aplikací IBM MQ classes for JMS , musí být obě aplikace schopny interpretovat data aplikace v těle zprávy stejným způsobem.

Z tohoto důvodu musí mít každý prvek dat aplikace zapsaný v těle zprávy aplikací XMS jeden z datových typů uvedených v souboru Tabulka 100 na stránce 655. Pro každý datový typ tabulka zobrazuje kompatibilní datový typ Java . Produkt XMS poskytuje metody pro zápis prvků dat aplikace pouze s těmito datovými typy.

Tabulka 100. Datové typy XMS , které jsou kompatibilní s datovými typy Java

XMS Datový typ	Představuje	Kompatibilní datový typ Java
System.Boolean	Logická hodnota true nebo false	typ boolean
System.Char16	Dvoubajtový znak	ZNAK
System.SByte	Podepsané 8bitové celé číslo	bajt
System.Int16	16bitové celé číslo se znaménkem	short
System.Int32	32bitové celé číslo se znaménkem	celé číslo
System.Int64	64bitové celé číslo se znaménkem	long
System.Float	Podepsané číslo s pohyblivou řádovou čárkou	float
System.Double	Číslo s pohyblivou řádovou čárkou a dvojitou přesností	dvojitý
System.String	Řetězec znaků	Řetězec

Informace o velikosti, maximální hodnotě a minimální hodnotě každého z těchto datových typů viz “XMS primitivní typy” na stránce 629.

Selektory zpráv

Aplikace XMS používá selektory zpráv k výběru zpráv, které chce přijmout.

Když aplikace vytvoří spotřebitele zpráv, může k němu přidružit výraz selektoru zpráv. Výraz selektoru zpráv určuje kritéria výběru.

Když se aplikace připojuje ke správci front IBM WebSphere MQ 7.0 , výběr zpráv se provádí na straně správce front. Produkt XMS neprovádí žádný výběr a jednoduše doručí zprávu, kterou obdržel od správce front, čímž poskytne lepší výkon.

Aplikace může vytvořit více než jednoho spotřebitele zpráv, každý s vlastním výrazem selektoru zpráv. Pokud příchozí zpráva splňuje kritéria výběru více než jednoho spotřebitele zpráv, produkt XMS doručí zprávu každému z těchto spotřebitelů.

Výraz selektoru zpráv může odkazovat na následující vlastnosti zprávy:

- Vlastnosti definované službou JMS
- IBM-definované vlastnosti
- Vlastnosti definované aplikací

Může také odkazovat na následující pole záhlaví zprávy:

- JMSCorrelationID
- JMSDeliveryMode
- JMSMessageID
- JMSPriority.
- JMSTimestamp
- JMSType.

Výraz selektoru zpráv však nemůže odkazovat na data v těle zprávy.

Zde je příklad výrazu selektoru zpráv:

```
JMSPriority > 3 AND manufacturer = 'Jaguar' AND model in ('xj6','xj12')
```

Produkt XMS doručí zprávu spotřebiteli zpráv s tímto výrazem selektoru zpráv pouze v případě, že má zpráva vyšší prioritu než 3; vlastnost definovaná aplikací, výrobce s hodnotou `Jaguar;` a jinou vlastnost definovanou aplikací, model s hodnotou `xj6` nebo `xj12.`

Pravidla syntaxe pro vytvoření výrazu selektoru zpráv v souboru XMS jsou stejná jako v souboru IBM MQ classes for JMS. Chcete-li získat informace o tom, jak sestavit výraz selektoru zpráv, prohlédněte si dokumentaci k produktu IBM MQ . Všimněte si, že ve výrazu selektoru zpráv musí být názvy vlastností JMS-defined názvy JMS a názvy vlastností IBM-defined musí být názvy IBM MQ classes for JMS . Ve výrazu selektoru zpráv nelze použít názvy XMS .

Mapování zpráv XMS na zprávy IBM MQ

Pole záhlaví JMS a vlastnosti zprávy XMS jsou mapovány na pole ve strukturách záhlaví zprávy IBM MQ .

Je-li aplikace XMS připojena ke správci front systému IBM MQ , jsou zprávy odeslané správci front mapovány na zprávy systému IBM MQ stejným způsobem, jako jsou zprávy systému IBM MQ classes for JMS mapovány na zprávy systému IBM MQ za podobných okolností.

Pokud je vlastnost XMSC_WMQ_TARGET_CLIENT cílového objektu nastavena na hodnotu XMSC_WMQ_TARGET_DEST_JMS, pole záhlaví JMS a vlastnosti zprávy odeslané do místa určení jsou mapovány na pole ve strukturách záhlaví MQMD a MQRFH2 zprávy IBM MQ . Nastavení vlastnosti XMSC_WMQ_TARGET_CLIENT tímto způsobem předpokládá, že aplikace, která obdrží zprávu, může zpracovat záhlaví MQRFH2 . Přijímající aplikací může být proto jiná aplikace XMS , aplikace IBM MQ classes for JMS nebo nativní aplikace IBM MQ , která byla navržena pro zpracování záhlaví MQRFH2 .

Je-li vlastnost XMSC_WMQ_TARGET_CLIENT cílového objektu nastavena na hodnotu XMSC_WMQ_TARGET_DEST_MQ, jsou pole záhlaví JMS a vlastnosti zprávy odeslané do místa určení mapovány na pole ve struktuře záhlaví MQMD zprávy IBM MQ . Zpráva neobsahuje záhlaví MQRFH2 a všechna pole a vlastnosti záhlaví JMS, které nelze mapovat na pole ve struktuře záhlaví MQMD, jsou ignorovány. Aplikace, která přijme zprávu, proto může být nativní IBM MQ , která není navržena pro zpracování záhlaví MQRFH2 .

Zprávy IBM MQ přijaté od správce front jsou mapovány na zprávy produktu XMS stejným způsobem jako zprávy produktu IBM MQ mapované na zprávy produktu IBM MQ classes for JMS za podobných okolností.

Pokud má příchozí zpráva IBM MQ záhlaví MQRFH2 , má výsledná zpráva XMS tělo, jehož typ je určen hodnotou vlastnosti **Msd** obsažené ve složce `mcd` záhlaví MQRFH2 . Pokud vlastnost **Msd** není přítomna v záhlaví MQRFH2 nebo pokud zpráva IBM MQ nemá záhlaví MQRFH2 , má výsledná zpráva XMS tělo, jehož typ je určen hodnotou pole *`Format`* v záhlaví MQMD. Pokud je pole *`Format`* nastaveno na MQFMT_STRING, zpráva XMS je textová zpráva. Jinak je zpráva XMS bajtovou zprávou. Pokud zpráva IBM

MQ neobsahuje záhlaví MQRFH2 , jsou nastavena pouze pole záhlaví JMS a vlastnosti, které lze odvodit z polí v záhlaví MQMD.

Další informace o mapování zpráv IBM MQ classes for JMS na zprávy IBM MQ viz “Mapování zpráv JMS na zprávy IBM MQ” na stránce 144.

Čtení a zápis deskriptoru zprávy z aplikace IBM MQ Message Service Client (XMS) for .NET

Můžete přistupovat ke všem polím deskriptoru zpráv (MQMD) zprávy IBM MQ s výjimkou StrucId a verze; BackoutCount lze číst, ale nelze do něj zapisovat.

Atributy zpráv poskytované aplikací IBM MQ Message Service Client (XMS) for .NET usnadňuje aplikacím XMS nastavení polí MQMD a také řízení aplikací IBM WebSphere MQ .

Při použití systému zpráv publikování/odběru platí některá omezení. Například pole MQMD, jako např. MsgID a CorrelId, jsou-li nastavena, jsou ignorována.

Funkce je také nedostupná, když je vlastnost **PROVIDERVERSION** nastavena na hodnotu 6.

Přístup k datům zprávy IBM MQ z aplikace IBM MQ Message Service Client (XMS) for .NET

K úplným datům zprávy IBM MQ včetně záhlaví MQRFH2 (je-li k dispozici) a dalších záhlaví IBM MQ (je-li k dispozici) v rámci aplikace IBM MQ Message Service Client (XMS) for .NET můžete přistupovat jako k tělu zprávy JMSBytesMessage.

Funkce popsaná v tomto tématu je k dispozici pouze v případě, že se připojujete ke správci front IBM WebSphere MQ 7.0 nebo novějšímu správci front a poskytovatel systému zpráv IBM MQ je v normálním režimu.

Vlastnosti cílového objektu určují, jak aplikace XMS přistupuje k celé zprávě IBM MQ (včetně záhlaví MQRFH2 , je-li k dispozici) jako k tělu zprávy JMSBytesMessage.

ALW Vývoj klientských aplikací AMQP

Podpora IBM MQ pro rozhraní API AMQP umožňuje administrátorovi systému IBM MQ vytvořit kanál AMQP. Při spuštění tento kanál definuje číslo portu, které přijímá připojení z klientských aplikací AMQP.

Kanál AMQP můžete nainstalovat na systémech AIX, Linux, and Windows ; není k dispozici na systémech IBM i nebo z/OS.

Klientská aplikace AMQP 1.0 se může připojit ke správci front pomocí kanálu AMQP.

Vývoj aplikací pomocí knihovny produktu Apache Qpid JMS

Úvod

Knihovna Apache Qpid JMS používá protokol AMQP 1.0 k poskytnutí implementace specifikace JMS 2.

Produkt Apache Qpid JMS používá některé aspekty protokolu AMQP 1.0 jiným způsobem než rozhraní API systému zpráv MQ Light . Produkt IBM MQ 9.2 přidává podporu pro kanály IBM MQ AMQP, aby se aplikace Apache Qpid JMS mohly připojit k produktu IBM MQ a provádět systém zpráv publikování/odběru, včetně použití sdílených odběrů.

V 9.3.0 Produkt IBM MQ 9.3 přidává další podporu pro kanály IBM MQ AMQP, aby se aplikace Apache Qpid JMS mohly připojit k produktu IBM MQ a provádět systém zpráv typu point-to-point. Další informace viz “Podpora dvoubodových kanálů AMQP” na stránce 662.

V 9.3.0 Produkt IBM MQ 9.3.0 přidává další podporu procházení fronty pro kanály IBM MQ AMQP, aby se aplikace Apache Qpid JMS mohly připojit k produktu IBM MQ a provádět procházení zpráv z fronty. Další informace viz “Podpora dvoubodových kanálů AMQP” na stránce 662.

V 9.3.0 Produkt IBM MQ 9.3.0 přidá dva další atributy kanálu pro kanály AMQP, TMPMODEL a TMPQPRFX. Tyto atributy jsou určeny pro modelovou frontu a předponu dočasné fronty, která se má použít při vytváření dočasné fronty.

Interkomunikace s ostatními IBM MQ aplikacemi

Je možné odesílat zprávy mezi aplikacemi Apache Qpid JMS a dalšími aplikacemi IBM MQ . Například aplikace Apache Qpid může publikovat zprávy v tématu a aplikace MQ Light je mohou přijmout vytvořením odběru.

Aplikace Apache Qpid JMS může také publikovat zprávy, které jsou spotřebovány tradičními aplikacemi IBM MQ , například pomocí volání MQSUB API pro přihlášení k odběru stejného tématu.

Podobně se mohou aplikace Apache Qpid JMS přihlásit k odběru témat IBM MQ , na kterých tradiční aplikace IBM MQ publikují zprávy.

Je také možné, aby aplikace Apache Qpid JMS sdílely odběr s aplikací MQ Light , pokud oba klienti určí stejný název sdílení a vzor tématu.

Všimněte si, že k tomu se aplikace Apache Qpid JMS nesmí připojit pomocí ID klienta. Tím je zajištěno, že název odběru IBM MQ používaný oběma aplikacemi je stejný.

Upozornění: Není možné, aby aplikace Apache Qpid JMS sdílely odběr s aplikací IBM MQ JMS .

Apache Qpid JMS

Podporovány jsou následující schopnosti produktu JMS :

- Režim potvrzení klienta, automatického potvrzení a dups ok (DUPS_OK_KVÉST)
  - Připojení s pověřeními nebo bez nich
  - Vytvoření spotřebitele v místě určení tématu
  - Vytvoření trvalého spotřebitele v místě určení tématu
  - Vytvoření sdíleného spotřebitele v místě určení tématu
  - Vytvoření sdíleného trvalého spotřebitele v místě určení tématu
  - Režimy potvrzení a automatického potvrzení klienta
  - Potvrzení zprávy a potvrzení relace
  - Zrušení odběru trvalého odběru
  - V 9.3.0 Vytvoření dočasné fronty
  - V 9.3.0 Vytvoření spotřebitele ve frontě nebo dočasném cíli fronty
  - V 9.3.0 MessageListeners rozhraní JMS
  - V 9.3.0 Spotřebitel JMS pro příjem těla; metoda JMS 2.0 s názvem `Consumer.receiveBody()`
  - V 9.3.0 Podporovány jsou následující typy zpráv JMS:
    - BytesMessage
    - MapMessage
    - ObjectMessage
    - StreamMessage
    - TextMessage
  - V 9.3.0 Procházení zpráv z fronty

Následující schopnosti JMS nejsou podporovány klienty AMQP:

- Použití transakčních relací a transakčních kontextů JMSContext
  - Použití selektorů zpráv
  - Použití atributu **nolocal**
  - Použití transakčních relací
  - Použití zpoždění doručení
  - V systému IBM MQ 9.3.0se jedná o procházení zpráv z fronty.
  - Vytvoření více trvalých odběrů nebo spotřebitelů se stejným ID klienta a tématem
  - V 9.3.0 Dočasná témata JMS
  - Filtry APMQP nejsou podporovány.

V 9.3.3 V produktu IBM MQ 9.3.3 se následující poznámka již nevztahuje na uživatele produktu Continuous Delivery .

Upozornění: V 9.3.0 Zákazník potvrzuje: pokud mají být použity případné nevyřešené přenosy zpráv AMQP, tj. pokud je vyžadováno potvrzení klienta o zprávách, musí se buď klienti včas usadit odesláním potvrzení zpráv v rámci včasného způsobu při použití režimu potvrzení klienta, nebo zvážit nastavení vlastnosti správce front MARKINT (**MsgMarkBrowseInterval**) na vyšší hodnotu.

Výchozí hodnota pro **MsgMarkBrowseInterval** je pět sekund. Pokud se aplikace neusadí v rámci této výchozí hodnoty, mohou se zobrazit duplicitní zprávy. Chcete-li se vyhnout duplikaci zpráv, musíte odpovídajícím způsobem zvýšit hodnotu **MsgMarkBrowseInterval** , ideálně ji nastavit na hodnotu NOLIMIT, aby představovala neomezený časový interval. Pokud dojde k havárii nebo odpojení aplikace před vyhodnocením zprávy, jsou zprávy zpřístupněny jiné aplikaci.

Další informace viz **MsgMarkBrowseInterval** . Vzhledem k tomu, že se jedná o vlastnost správce front, bude hodnota, kterou jste nastavili, použita pro všechny aplikace připojené k tomuto správci front.

V AMQP je **MsgMarkBrowseInterval** platný pouze pro fronty, ne pro odběry.

Stažení ukázkových klientů AMQP

Produkt IBM MQ nedodává klienty AMQP, ale můžete stáhnout klienty MQ Light nebo klienty AMQP s otevřeným zdrojovým kódem na základě knihoven Apache Qpid. Další informace viz IBM MQ Light a Apache Qpid.

Můžete si také stáhnout další klienty AMQP s otevřeným zdrojovým kódem na základě knihoven Qpid Apache . Další informace viz téma https://qpid.apache.org/index.html.

Upozornění: Podpora IBM nemůže poskytnout podporu konfigurace nebo defektu pro tyto balíky klienta a jakékoli otázky týkající se použití nebo sestavy defektů kódu by měly být směrovány na příslušné projekty.

Implementace klientů AMQP na IBM MQ

Když je aplikace připravena k implementaci, vyžaduje všechny funkce monitorování, spolehlivosti a zabezpečení ostatních podnikových aplikací. Může si také vyměňovat data s jinými podnikovými aplikacemi.

Když implementujete klienta AMQP, můžete si vyměňovat zprávy s aplikacemi IBM MQ . Pokud například použijete klienta AMQP k odeslání zprávy řetězce JavaScript , aplikace IBM MQ obdrží zprávu MQ , kde je pole formátu MQMD nastaveno na MQSTR.

Správa kanálu AMQP

Kanál AMQP lze spravovat stejným způsobem jako jiné kanály MQ . Můžete použít příkazy MQSC, zprávy příkazů PCF nebo IBM MQ Explorer k definování, spuštění, zastavení a správě kanálů. V části Vytvoření a použití kanálů AMQPjsou uvedeny ukázkové příkazy pro definování a spuštění připojení klientů ke správci front.

Když je spuštěn kanál AMQP, můžete jej otestovat připojením klienta AMQP 1.0 . Například MQ Light, Apache Qpid Proton nebo Apache Qpid JMS.

Související úlohy

Vytváření a používání kanálů AMQP

Zabezpečení klientů AMQP

ALW MQ Light, Apache Qpid JMSa AMQP (Advanced Message Queuing Protocol).

Klienti MQ Light , Apache Qpid klienti jako Apache Proton a Apache Qpid JMS API jsou založeny na spojovém protokolu OASIS Standard AMQP 1.0 . AMQP určuje způsob odesílání zpráv mezi odesilateli a příjemci. Aplikace vystupuje jako odesilatel, když aplikace odesílá zprávu zprostředkovateli zpráv, například IBM MQ. Produkt IBM MQ vystupuje jako odesílatel, když odesílá zprávu do aplikace AMQP.

Některé z výhod AMQP jsou následující:

- Otevřený standardizovaný protokol
- Kompatibilita s ostatními klienty typu open source AMQP 1.0
- K dispozici je mnoho implementací klienta typu open source.

Ačkoli se kterýkoli klient AMQP 1.0 může připojit ke kanálu AMQP, některé funkce AMQP nejsou podporovány, například transakce nebo vícenásobné relace.

Další informace viz AMQP.org web a OASIS Standard AMQP 1.0 PDF.

Rozhraní API MQ Light a Apache Qpid JMS mají následující funkce systému zpráv:

- Doručení zprávy nanejvýš jednou
- Alespoň jednou doručení zprávy
- Adresování místa určení řetězce tématu
- Trvanlivost zprávy a místa určení
- Sdílené cíle, které umožňují více odběratelům sdílet pracovní zátěž
- Převzetí klienta pro snadné vyřešení zablokovaných klientů
- Konfigurovatelné čtení před zprávami
- Konfigurovatelné potvrzení zpráv

Úplnou dokumentaci k rozhraní API Apache Qpid JMS naleznete v části Qpid JMS.

Související úlohy

Vytváření a používání kanálů AMQP

Zabezpečení klientů AMQP

ALW Podpora produktu AMQP 1.0

Kanály AMQP poskytují úroveň podpory pro aplikace AMQP 1.0-compliant .

Kanály AMQP podporují podmnožinu protokolu AMQP 1.0 . Klienty kompatibilní s AMQP 1.0 můžete připojit ke kanálu IBM MQ AMQP. Chcete-li používat všechny funkce systému zpráv podporované kanály AMQP, musíte správně nastavit hodnotu určitých polí AMQP 1.0 .

Tyto informace nastiňují způsob, jakým musí být pole AMQP formátována, a uvádějí funkce specifikace AMQP 1.0 , které kanály AMQP nepodporují.

Následující funkce specifikace AMQP 1.0 buď nejsou podporovány, nebo jsou omezeny v jejich použití:

Rámec ATTACH

V 9.3.0

Kanály AMQP očekávají, že schopnosti v rámci ATTACH budou obsahovat jednu z následujících možností:

```
topic
temporary queue
queue
shared
```

Schopnosti implikují typ objektu a v případě více schopností je pořadí priority výběru schopnosti téma, dočasná fronta, fronta.

Pokud schopnost neobsahuje očekávanou hodnotu, je výchozí schopností téma. Všechny ostatní schopnosti jsou ignorovány.

Poznámka: Někteří klienti AMQP nenastavují tyto schopnosti a získají výchozí chování IBM MQ publikování/odběru. Například konektor Quarkus Reactive Messaging AMQP 1.0 nastavuje schopnosti pouze od verze 2.8.0CR1 dále.

V 9.3.0 Kanály AMQP očekávají, že `distribution-Mode` v rámci ATTACH bude obsahovat jednu z následujících položek pro zdroj nebo cíl:

- `Přesunout`
- `kopie`

kde `move` znamená destruktivní získání a `copy` znamená prohlížeč.

Poznámka: Není-li parametr `distribution-Mode` nastaven nebo je-li nastaven na jinou hodnotu než `copy`, předpokládá se `move` .

Názvy odkazů

Kanály AMQP očekávají, že název odkazu AMQP bude postupovat podle jednoho z pěti formátů:

- Prosté téma (pro publikování a odběr)
  - Publikování zpráv: Prostý řetězec tématu (například název odkazu "`/sports/football`") způsobí, že zpráva bude publikována v tématu `/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)
  - Řetězec tématu s komentářem, který popisuje soukromý odběr ve formátu: "`private:`*`topic string`*" (například: "`private:/sports/football`"). Chování je identické s prostým řetězcem tématu. Deklarace `private` odlišuje odběr specifický pro konkrétního klienta AMQP od odběru sdíleného mezi klienty.
- Sdílené téma s komentářem (pro přihlášení k odběru)
  - Řetězec tématu s komentářem, který popisuje sdílený odběr ve formátu: "`share:`*`share name:topic string`*" (například: "`share:bbc:/sports/football`").
- V 9.3.0 Fronta (pro systém zpráv typu point-to-point pro producenta a spotřebitele)
  - Producent odesílá zprávy; řetězec názvu fronty způsobí, že producent odešle zprávu ve frontě.
  - Spotřebitel přijímá zprávy; řetězec názvu fronty způsobí, že spotřebitel přijme zprávy z fronty.
- V 9.3.0 Prázdné (pro systém zpráv typu point-to-point v dočasné frontě)
  - Producent odešle zprávy v dočasné frontě; prázdné způsobí, že producent odešle zprávu v dočasné frontě.

– Spotřebitel přijímá zprávy v dočasné frontě. Prázdná volba způsobí, že spotřebitel bude přijímat zprávy z dočasné fronty.

Další informace o způsobu mapování zpráv AMQP na zprávy IBM MQ a z nich viz “Mapování polí AMQP na pole IBM MQ (příchozí zprávy)” na stránce 666.

Maximální délka řetězců témat, názvů sdílení a ID klientů

Řetězec tématu, název sdílení a ID klienta musí být v rozsahu 10237 bajtů. 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 sdílení je krátký
- dlouhý název sdílení, ale krátký řetězec tématu

ID kontejnerů

Kanály AMQP očekávají, že ID kontejneru performativu AMQP Open bude obsahovat jedinečné ID klienta AMQP. Maximální délka ID klienta AMQP je 256 znaků a ID může obsahovat alfanumerické znaky, znak procenta (%), 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 připojení AMQP, který se pokouší koordinovat novou transakci nebo rámec přenosu AMQP, který se pokouší deklarovat novou transakci, je odmítnut s chybovou zprávou.

Stav doručení

Kanály AMQP podporují pouze stav doručení pro dispoziční rámce Přijato, Uvolněno nebo Změněno. Všimněte si, že tam, kde se používá stav Změněno, kanály AMQP nepodporují nedoručitelnou volbu-zde.

Související úlohy

Vytváření a používání kanálů AMQP

Zabezpečení klientů AMQP

V 9.3.0 ALW Podpora dvoubodových kanálů AMQP

Kanál IBM MQ AMQP poskytuje podporu pro odesílání zpráv do front a příjem zpráv z front.

Klienti AMQP, jako například Apache Qpid™ JMS , si při odesílání rámce připojení AMQP vyžádají funkci `queue` nebo `temporary-queue` . Funkce umožňují kanálu AMQP identifikovat objekt jako frontu, dočasnou frontu nebo téma. Při absenci možnosti fronty nebo dočasné fronty, nebo dokonce jakékoli schopnosti, se předpokládá, že požadavek je pro téma.

Kanály IBM MQ AMQP poskytují podporu typů front pro následující:

Přijmout a odeslat frontu
Zprávy lze odeslat do fronty a spotřebovat z fronty. Pro příjem zpráv jsou podporovány synchronní i asynchronní režimy.

V 9.3.0 **Zpráva procházení fronty**
Kromě vkládání zpráv do fronty a získávání zpráv z fronty lze zprávy také procházet z fronty.

Podpora dočasných front
Zprávy lze odeslat do dočasné fronty a spotřebovat z dočasné fronty. Všimněte si, že dočasné odstranění fronty je podporováno, pokud je k odstranění dočasné fronty použit stejný dočasný objekt fronty, který byl použit k vytvoření dočasné fronty.

Systém SYSTEM.DEFAULT.MODEL.QUEUE se používá při vytváření dočasné fronty a předpona pro dočasnou frontu bude `AMQP.*`.

Systém SYSTEM.DEFAULT.MODEL.QUEUE je standardně dočasná dynamická fronta, ale můžete použít vlastnost **`Definition type`** v SYSTEM.DEFAULT.MODEL.QUEUE pro změnu fronty na trvalou dynamickou frontu.

permanentní dynamická fronta
Trvalá dynamická fronta je odstraněna, když klient AMQP, například knihovna Apache Qpid JMS , odešle požadavek s rámem `detach` s atributem **`closed`** nastaveným na hodnotu *true*.

Důležité:

Chování Qpid JMS :
Musíte volat příkaz rozhraní API Qpid JMS , například metodu `javax.jms.TemporaryQueue.delete()` pro zničení fronty po použití, a tento proces také vymaže zprávy přítomné ve frontě.

Pokud takový příkaz nezadáte, zůstane fronta po zavření připojení stále přítomna se všemi zprávami.

-

dočasná dynamická fronta
Dočasná dynamická fronta se odstraní, když klient AMQP uzavře připojení.

Důležité:

Chování Qpid JMS :
Pokud zavoláte příkaz rozhraní API Qpid JMS , například metodu `javax.jms.TemporaryQueue.delete()` , zavřete připojení JMS nebo přerušíte připojení, fronta se odstraní a všechny zprávy se ztratí.

Zavření relace JMS samo o sobě nezpůsobí odstranění dočasné fronty, přestože dočasná fronta mohla být vytvořena pomocí metody `javax.jms.Session.createTemporaryQueue()` .

-

Související úlohy
Vytváření a používání kanálů AMQP
Zabezpečení klientů AMQP

ALW Mapování polí zpráv AMQP a IBM MQ

Zprávy AMQP se skládají ze záhlaví, anotací doručení, anotací zpráv, vlastností, vlastností aplikace, těla a zápatí.

Zprávy APMQP se skládají z následujících částí:

Header
Volitelné záhlaví obsahuje pět pevných atributů zprávy:

- **trvanlivý** -Určuje požadavky na trvanlivost.
- **priority** -Relativní priorita zprávy.
- **ttl** -doba platnosti v milisekundách
- **first-acquirer** -Pokud je hodnota true, zpráva nebyla získána žádným jiným odkazem.
- **delivery-count** -počet předchozích neúspěšných pokusů o doručení.

Doručení-anotace
Volitelné. Určuje nestandardní atributy záhlaví zprávy pro různé zamýšlené příjemce. Anotace doručení předávají informace z odesílajícího rovnocenného uzlu přijímajícímu rovnocennému partnerovi.

Anotace zpráv
Volitelné. Určuje nestandardní atributy záhlaví zprávy pro různé zamýšlené příjemce. Sekce anotací 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 ekvivalentní deskriptoru zprávy MQ . Obsahuje následující pevná pole:

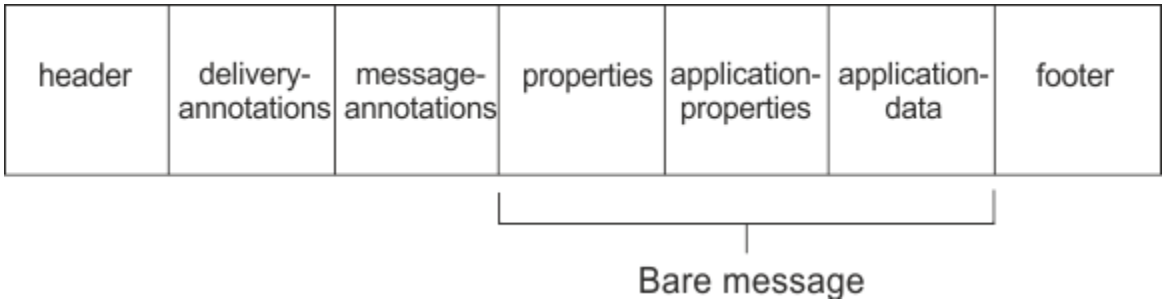
- **message-id** -Identifikátor zprávy aplikace
- **user-id** -ID uživatele pro vytváření
- **to** -adresa uzlu, pro který je zpráva určena.
- **subject** -Předmět zprávy.
- **reply-to** -uzel, na který odesílají odpovědi.
- **correlation-id** -Identifikátor korelace aplikace.
- **content-type** -Typ obsahu MIME
- **content-encoding** -typ obsahu MIME. Používá se jako modifikátor typu obsahu.
- **absolute-expiry-time** -čas, kdy byla tato zpráva považována za vypršelou.
- **creation-time** -čas, kdy byla tato zpráva vytvořena
- **group-id** -skupina, do které tato zpráva patří.
- **group-sequence** -Pořadové číslo této zprávy v rámci skupiny.
- **reply-to-group-id** -skupina, do které zpráva odpovědi patří.

Aplikace-vlastnosti
Ekvivalentní k vlastnostem zprávy MQ .

Tělo
Ekvivalent informačního obsahu uživatele produktu MQ .

Zápatí
Volitelné. Zápatí se používá pro podrobnosti o zprávě nebo doručení, které lze vypočítat nebo vyhodnotit pouze po vytvoření nebo zobrazení celé holé zprávy (například hašování zpráv, HMAC, podpisy a podrobnosti šifrování).

Formát zprávy AMQP je znázorněn na následujícím obrázku:



Část properties, application-properties a application-data se nazývá "holá zpráva". Jedná se o zprávu odeslanou odesilatelem, která je neměnná. Příjemce vidí celou zprávu včetně záhlaví, zápatí, anotací doručení a anotací zpráv.

Úplný popis formátu zprávy AMQP 1.0 viz OASIS Standard na adrese https://docs.oasis-open.org/amqp/core/v1.0/amqp-core-complete-v1.0.pdf.

Související úlohy
Vytváření a používání kanálů AMQP

ALW Mapování polí IBM MQ na pole AMQP (odchozí zprávy)

Když je zpráva IBM MQ publikována a produkt IBM MQ ji odešle spotřebiteli 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 jinou než výchozí hodnotou. Pět polí záhlaví je na počátku odvozeno od ekvivalentní vlastnosti `mq_amqp.Hdr` , pokud je nastavena, a poté upraveno, jak ukazuje následující tabulka:

Tabulka 101. Mapování polí záhlaví

Pole	Výchozí hodnota	Hodnota
Trvalý	ne	Hodnota true, pokud je parametr `MQMD.Persistence` nastaven na hodnotu `MQPER_PERSISTENT`, jinak na hodnotu false.
priorita	4	Z `mq_amqp.Hdr.Pri`, je-li nastaveno, nebo jinak z `MQMD.Priority`, je-li nastaveno. Není-li nastaveno, nastavte na hodnotu 4.
ttl	Není k dispozici.	`MQMD.Expiry` v milisekundách. Je-li hodnota parametru `MQMD.Expiry` MQEI_UNLIMITED, nastavte maximální hodnotu pro pole AMQP ttl.
první nabyvatel	ne	Z `mq_amqp.Hdr.Fac`, je-li nastaveno, nebo jinak false.
Počet doručení	0	Z `mq_amqp.Hdr.Dct`, je-li nastaveno, nebo 0 jinak.

anotace-doručení

Nastavte podle potřeby kanálem AMQP.

anotace-zprávy

Není součástí balení.

Vlastnosti

Vlastnosti budou neupravené z ekvivalentních vlastností `mq_amqp.Prp` , pokud jsou nastaveny. Pokud zpráva původně nebyla zprávou AMQP (tj. typ PutApplnení MQAT_AMQP), vygeneruje se sekce vlastností podle popisu v následující tabulce:

Tabulka 102. Mapování polí vlastností

Název	Hodnota
id-zprávy	Parametr `MQMD.MsgId` je nastaven jako binární.
id-uživatele	Formát UTF-8 souboru `MQMD.UserIdentifier` je nastaven jako binární v pořadí bajtů sítě.
do	Fronta, ze které byla zpráva získána, nebo v případě publikování řetězec tématu.

Tabulka 102. Mapování polí vlastností (pokračování)	
Název	**Hodnota**
předmět	Není nastaveno.
odpovědět-na	Parametr `MQMD.ReplyToQ` není-li prázdný, jinak není nastaven.
ID korelace	Parametr `MQMD.CorrelId` je nastaven jako binární, pokud není prázdný, jinak není nastaven.
Content-Type	Není nastaveno.
kódování obsahu	Není nastaveno.
absolute-expirační-time	Není nastaveno.
čas vytvoření	Pole `MQMD.PutDate` a `MQMD.PutTime` se používají ke generování časového razítka.
id-skupiny	Není nastaveno.
pořadí skupin	Není nastaveno.
id-skupiny-odpovědi	Není nastaveno.

vlastnosti aplikace

Všechny vlastnosti IBM MQ ve skupině "usr" jsou přidány jako **application-properties**.

tělo

Kanál AMQP provede příkaz get s převedením, aby převedl informační obsah IBM MQ do formátu UTF-8.

Pokud informační obsah IBM MQ neobsahuje zprávu AMQP, pak je informační obsah IBM MQ nastaven v těle jako jedna sekce řetězcových dat pro formát MQFMT_STRING (poskytnutý převod na UTF-8 byl úspěšný), nebo jako jedna sekce binárních dat jinak.

Pokud je zahrnuta zpráva ve formátu AMQP, pak je nastavena jako tělo. Jakákoli záhlaví IBM MQ (bez vlastností zpráv, které jsou vráceny v popisovači zprávy), která předcházejí zprávě AMQP, se předřadí jako binární hodnota, pokud je tělo posloupností AMQP. Jinak se záhlaví IBM MQ vyřadí.

zápatí

Není zahrnuto žádné zápatí.

Související úlohy
Vytváření a používání kanálů AMQP
Zabezpečení klientů AMQP
Související odkazy
MQMD-Deskriptor zpráv

ALW Mapování polí AMQP na pole IBM MQ (příchozí zprávy)

Když kanál AMQP přijme zprávu a vloží ji do souboru IBM MQ, rozšíří některé atributy zprávy AMQP do ekvivalentních atributů zprávy IBM MQ .

Při mapování příchozí zprávy AMQP platí následující omezení:

- Pokud je pole `message-id` nebo `correlation-id` v části vlastností uuid nebo ulong, pak je zpráva odmítnuta.
- Jakýkoli `message-annotations` způsobí, že zpráva bude odmítnuta.
- Sekce `delivery-annotations` a `footer` jsou povoleny, ale nejsou šířeny do zprávy IBM MQ .

Následující podsekce zobrazují výraz IBM MQ zprávy AMQP.

deskriptor zprávy

Tabulka 103. Deskriptor zprávy pro zprávu AMQP

Pole	**Hodnota**
StrucId	ID_STRUC_MQMD_STRUC_ID
Verze	MQMD_VERSION_1
Sestava	MQRO_NONE
MsgType	MQMT_DATAGRAM
Vypršení	Hodnota převzatá z pole `ttl` v záhlaví zprávy AMQP
Zpětná vazba	MQFB_NONE
Kódování	MQENC_NORMAL
CodedCharSetId	1208 (UTF-8)
Formát	Viz informační obsah
Priorita	Hodnota převzatá z pole `priority` v záhlaví zprávy AMQP. Je-li nastaveno, je omezeno na maximálně 9. Není-li nastaveno, použije se výchozí hodnota 4.
Trvání	Pokud je 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 přidělí jedinečný 24bajtový identifikátor MsgId.
Korrelld	Hodnota převzatá z pole `correlation-id` ve vlastnostech AMQP, je-li nastavena. Nastavte na 24bajtovou binární hodnotu. Jinak nastavte na MQCI_NONE/.
BackoutCount	0
ReplyToQ	V 9.3.0 Hodnota převzatá z pole `reply-to` ve vlastnostech AMQP, je-li nastavena. Jinak nastavte na "".
ReplyToQMgr	""
V 9.3.0 Sestava	Hodnota odvozená z libovolných vlastností sestavy JMS IBM nastavených ve vlastnostech aplikace AMQP.
UserIdentifier	Nastavit na identifikátor ověřeného uživatele, který se připojil ke kanálu AMQP
AccountingToken	MQACT_NONE
ApplIdentityData	Hexadecimální řetězec. Nastavte na posledních 8 bajtů identifikátoru připojení MQ kanálu AMQP.
PutApplType	MQAT_AMQP
PutApplName	
PutDate	Hodnota převzatá z pole `creation-time` vlastností AMQP, je-li nastavena. Jinak nastavte na aktuální datum.
PutTime	Hodnota převzatá z pole `creation-time` vlastností AMQP, je-li nastavena. Jinak nastavte na aktuální čas.

Tabulka 103. Deskriptor zprávy pro zprávu AMQP (pokračování)	
Pole	**Hodnota**
ApplOriginData	""

Vlastnosti zprávy

Existují dva důvody pro nastavení vlastností zprávy:

- Chcete-li povolit, aby části zprávy AMQP protékly správcem front bez ovlivnění informačního obsahu zprávy.
- Chcete-li povolit výběr volby `application-properties`.

Následující tabulka zobrazuje vlastnosti, které jsou nastaveny ze zprávy AMQP:

Tabulka 104. Vlastnosti zprávy AMQP			
Název vlastnosti	**MQRFH2 název**	**Typ**	**Popis**
AMQPListener	mq_amqp.Lis	MQTYPE_STRING	Identifikační řetězec pro kanál AMQP. Používá se ke generování zprávy, aby zainteresované strany mohly zjistit, která verze zprávu vložila (například servisní tým při diagnostice problémů). Hodnota není ověřena správcem front a nesmí být dokumentována externě.
AMQPVersion	mq_amqp.Ver	MQTYPE_STRING	Verze zprávy AMQP. Není-li uveden, předpokládá se "1.0". Hodnota není ověřena správcem front.
AMQPClient	mq_amqp.Cli	MQTYPE_STRING	Identifikační řetězec pro rozhraní API. Používá se k odeslání zprávy AMQP do kanálu, aby zúčastněné strany mohly zjistit, která verze zprávu vložila (například servisní tým při diagnostice problémů). Hodnota není ověřena správcem front a nesmí být dokumentována externě.
AMQPDurable	mq_amqp.Hdr.Dur	MQTYPE_BOOLEAN	Hodnota pole `durable` v záhlaví zprávy AMQP, je-li nastavena.
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	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.

Tabulka 104. Vlastnosti zprávy AMQP (pokračování)			
Název vlastnosti	**MQRFH2 název**	**Typ**	**Popis**
AMQPMsgId	mq_amqp.Prp.Mid	MQTYPE_STRING	Hodnota pole `message-id` ve vlastnostech AMQP, je-li nastavena jako řetězec.
		MQTYPE_BYTE_STRING	Hodnota pole `message-id` ve vlastnostech AMQP, je-li nastavena jako bajtový řetězec.
AMQPUserId	mq_amqp.Prp.Uid	MQTYPE_BYTE_STRING	Hodnota pole `user-id` ve vlastnostech AMQP, je-li nastavena.
AMQPTo-Přejít na	mq_amqp.Prp.To	MQTYPE_STRING	Hodnota pole `to` ve vlastnostech AMQP, je-li nastavena.
Předmět AMQPSubject	mq_amqp.Prp.Sub	MQTYPE_STRING	Hodnota pole `subject` ve vlastnostech AMQP, je-li nastavena.
AMQPReplyTo	mq_amqp.Prp.Rto	MQTYPE_STRING	Hodnota pole `reply-to` ve vlastnostech AMQP, je-li nastavena.
AMQPCorrelationId	mq_amqp.Prp.Cid	MQTYPE_STRING	Hodnota pole `correlation-id` ve vlastnostech AMQP, je-li nastavena jako řetězec.
		MQTYPE_BYTE_STRING	Hodnota pole `correlation-id` ve vlastnostech AMQP, je-li nastavena jako bajtový řetězec.
AMQPContentType	mq_amqp.Prp.Cnt	MQTYPE_STRING	Hodnota pole `content-type` ve vlastnostech AMQP, je-li nastavena.
AMQPContentEncoding	mq_amqp.Prp.Cneing	MQTYPE_STRING	Hodnota pole `content-encoding` ve vlastnostech AMQP, je-li nastavena.
AMQPAbsoluteExpiryČas	mq_amqp.Prp.Aet	MQTYPE_STRING	Hodnota pole `absolute-expiry-time` ve vlastnostech AMQP, je-li nastavena.
AMQPCreationTime	mq_amqp.Prp.Crt	MQTYPE_STRING	Hodnota pole `creation-time` ve vlastnostech AMQP, je-li nastavena.
AMQPGroupId	mq_amqp.Prp.Gid	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	MQTYPE_STRING	Hodnota pole `reply-to-group-id` ve vlastnostech AMQP, je-li nastavena.

Každá vlastnost aplikace ze zprávy AMQP je nastavena jako vlastnost zprávy IBM MQ . Část `application-properties` musí být rekonstituována stejně jako bajt pro bajt, a proto platí následující omezení:

- Je-li vlastnost aplikace odmítnuta ověřovacím kódem MQSETMP, bude zpráva odmítnuta. Příklad:
  - Délka názvu vlastnosti je omezena na hodnotu MQ_MAX_PROPERTY_NAME_LENGTH.

  - Název vlastnosti musí odpovídat pravidlům definovaným ve specifikaci jazyka Java pro identifikátory Java .
  - Název vlastnosti nesmí začínat na `JMS` nebo `usr.JMS` s výjimkou dokumentovaných vlastností JMS, které lze nastavit.
  - Název vlastnosti nesmí být klíčové slovo SQL.
- Vlastnost aplikace obsahující znak Unicode U+002E (".") způsobí, že zpráva bude odmítnuta. Vlastnost musí být vyjádřitelná ve skupině vlastností "usr" používané JMS.
- Podporovány jsou pouze vlastnosti null, boolean, byte, short, int, long, float, double, binary a string. Vlastnost aplikace s jakýmkoli jiným typem způsobí, že zpráva bude odmítnuta.

V 9.3.0 Pomocí parametru `application-properties`můžete nastavit následující vlastnosti rozhraní JMS:

- VÝJIMKA-JMS_IBM_REPORT_EXCEPTION
- JMS_IBM_REPORT_EXPIRATION
- JMS_IBM_REPORT_COA
- JMS_IBM_REPORT_COD
- JMS_IBM_REPORT_PAN
- JMS_IBM_REPORT_NAN
- JMS_IBM_REPORT_PASS_MSG_ID
- JMS_IBM_REPORT_PASS_CORREL_ID
- JMS_IBM_REPORT_DISCARD_MSG

Všimněte si, že názvy a hodnoty vlastností jsou konzistentní s ekvivalentními podrobnostmi “Mapování JMS polí specifických pro poskytovatele” na stránce 155 a že hodnoty, které nejsou platné, jsou ignorovány.

informační obsah

- Pro AMQP body s jednou sekcí binárních dat jsou binární data (kromě bitů AMQP) vložena jako informační obsah IBM MQ s formátem MQFMT_NONE.
- Pro AMQP body s jednou sekcí řetězcových dat jsou řetězcová data (kromě bitů AMQP) vložena jako informační obsah IBM MQ s formátem MQFMT_STRING.
- Jinak AMQP body vytvoří informační obsah tak, jak je, s formátem MQFMT_AMQP.

Související úlohy

Vytváření a používání kanálů AMQP

Zabezpečení klientů AMQP

ALW Spolehlivost doručení zpráv

Tento oddíl porovnává funkce spolehlivosti pro rozhraní MQ Light API a Apache Qpid JMS.

Související úlohy

Vytváření a používání kanálů AMQP

Zabezpečení klientů AMQP

ALW Spolehlivost zpráv produktu MQ Light

Existují čtyři funkce rozhraní API MQ Light , které vám umožňují řídit spolehlivost doručování zpráv do a z aplikací AMQP.

Patří mezi ně:

- “Kvalita zpráv služby (QOS)” na stránce 671
- “Automatické potvrzení odběratele” na stránce 671

- “Doba platnosti předplatného” na stránce 672
- “Trvalost zpráv” na stránce 672

Kvalita zpráv služby (QOS)

MQ Light API nabízí dvě kvality služeb:

- Nanejvýš jednou
- Alespoň jednou

Můžete si vybrat, jakou kvalitu služeb mají vydavatelé a odběratelé používat.

Používáte-li klienta MQ Light , nastavte volbu klienta nebo odběr **qos** na hodnotu *QOS_AT_MOST_ONCE* nebo *QOS_AT_LEAST_ONCE*.

Používáte-li jiného klienta AMQP, nastavte atribut **`settled`** rámce přenosu (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 volbu **QOS** *0* (nejvýše jednou), vydavatel nečeká na potvrzení od správce front, než zruší svou kopii zprávy.

Pokud připojení ke správci front selže před dokončením odeslání, nemusí odběratelé zprávu přijmout.

Pokud vydavatel zvolí volbu **QOS** *1* (alespoň jednou), vyčká vydavatel na potvrzení správce front, že zpráva byla zapsána do front odběratele, než zruší svou kopii zprávy.

Dojde-li během odesílání k selhání připojení ke správci front, vydavatel znovu odešle zprávu po opětovném připojení ke správci front.

přihlášení odběru

Pokud odběratel zvolí volbu **QOS** *0* , správce front nečeká na potvrzení od odběratele před zrušením jeho kopie zprávy.

Pokud připojení k odběrateli selže dříve, než odběratel obdrží zprávu, může být tato zpráva ztracena.

Pokud odběratel zvolí volbu **QOS** *1* , správce front čeká na potvrzení od odběratele před zrušením jeho kopie zprávy. V 9.3.3 Z IBM MQ 9.3.3, potvrzené zprávy jsou odebrány v dávkách, aby se zlepšil výkon. Další informace uvádí téma “Odebrání potvrzených zpráv AMQP z fronty v dávkách” na stránce 673.

Pokud připojení k odběrateli selže dříve, než odběratel obdrží zprávu, je zpráva uchovávána správcem front. Správce front znovu odešle zprávu odběrateli, když se správce front znovu připojí, nebo jinému odběrateli, pokud je odběr sdílený.

Automatické potvrzení odběratele

Pokud odběratel zvolí volbu **QOS** *1* (alespoň jednou), musí potvrdit příjem každé zprávy předtím, než správce front zruší svou kopii. Odběratel se může rozhodnout, kdy bude zprávy potvrzovat.

Je-li volba **`auto-confirm`** nastavena na hodnotu *true*, klient MQ Light automaticky potvrdí doručení každé zprávy po úspěšném přijetí zprávy přes síť.

Tím zajistíte, že pokud dojde k selhání sítě, bude zpráva znovu doručena do aplikace. Je však stále možné, aby aplikace ztratila zprávu, pokud dojde k selhání aplikace mezi klientem MQ Light , který zprávu potvrzuje, a aplikací, která ji zpracovává.

Je-li volba **`auto-confirm`** nastavena na hodnotu *false*, klient MQ Light automaticky nepotvrdí doručení zprávy, ale ponechá ji na aplikaci, aby rozhodla, kdy má být potvrzena.

To umožňuje aplikaci provést aktualizaci externího prostředku, například databáze nebo souboru, před potvrzením správci front, že zpráva byla nyní zpracována a lze ji zrušit.

Doba platnosti předplatného

Když se aplikace přihlásí k odběru, rozhodne, zda odběr a místo určení, kde jsou pro daný odběr uloženy zprávy, budou nadále existovat i po odpojení aplikace.

Volba MQ Light přihlásit se k odběru **ttl** se používá k určení doby (v milisekundách), po kterou odběr nadále existuje po odpojení aplikace. Pokud se aplikace před touto dobou znovu připojí, bude odběr obnoven a aplikace bude moci nadále přijímat zprávy z tohoto odběru.

Pokud doba platnosti uplyne bez opětovného připojení aplikace, odběr bude odebrán a všechny zprávy uložené v místě určení budou ztraceny, a to i v případě, že se jedná o trvalé zprávy.

Pokud je důležité, abyste neztratili zprávy, musíte zadat hodnotu doby platnosti pro aplikaci, která je dostatečně vysoká, aby se zajistilo, že zprávy nebudou během výpadku ztraceny.

Trvalost zpráv

Perzistence zpráv je řízena aplikacemi publikování a odběru a konfigurací objektů tématu IBM MQ .

Pokud odběratel AMQP používá **QOS** *0* (nejvýše jednou) a vytvoří dočasný odběr, kanál AMQP vždy vloží dočasné zprávy do fronty odběratele bez ohledu na další volby popsané v následujícím textu.

Uvědomte si, že pokud je správce front zastaven, dojde ke ztrátě odběru i zpráv.

Pokud vydavatel AMQP nastaví záhlaví AMQP **durable** na hodnotu *true*, kanál AMQP vloží trvalé zprávy do front odběratele.

Je-li správce front z nějakého důvodu zastaven, budou zprávy i po restartování správce front nadále k dispozici pro odběratele.

Není-li záhlaví **durable** nastaveno, kanál AMQP zvolí perzistenci publikovaných zpráv na základě atributu **DEFPSIST** příslušného objektu tématu IBM MQ .

Standardně se jedná o SYSTEM.BASE.TOPIC, který používá atribut **DEFPSIST** s hodnotou *NO* (dočasný).

Upozornění: Novější verze klienta MQ Light nepodporují nastavení trvalého záhlaví AMQP.

Související úlohy

Vytváření a používání kanálů AMQP

Zabezpečení klientů AMQP

ALW Spolehlivost zpráv Apache Qpid JMS

Existují čtyři funkce knihovny Apache Qpid™ JMS , které vám umožňují řídit spolehlivost doručování zpráv do a z aplikací AMQP.

Jedná se o:

- “Publikování” na stránce 673 V 9.3.0 /Producent pro systém zpráv typu point-to-point
  - Vypršení platnosti zprávy
  - Trvalost zpráv
- “přihlášení odběru” na stránce 673
  - Trvanlivost předplatného
  - Režim potvrzení relace V 9.3.0 (Použitelné také pro systém zpráv typu point-to-point spotřebitele)

Publikování

Vypršení platnosti zprávy

Nastavení hodnoty doby platnosti producenta JMS ovlivní dobu vypršení platnosti poskytnutou zprávám publikovaným tímto producentem zpráv.

Ujistěte se, že hodnota doby platnosti pro producent JMS je dostatečně velká, aby byly zprávy spotřebovány před vypršením jejich platnosti.

Alternativně ponechání nenastavené hodnoty doby platnosti zabrání vypršení platnosti zprávy z fronty odběrů.

Trvalost zpráv

Nastavení režimu doručení producenta zpráv JMS nastaví perzistenci zprávy IBM MQ publikované na určené téma.

Ujistěte se, že používáte **`DeliveryMode`**.PERSISTENT pro zprávy, které musí být uchovány při ukončení správce front nebo při výpadku.

přihlášení odběru

Trvanlivost předplatného

Kanály AMQP podporují vytváření trvalých odběrů pomocí trvalých verzí metod spotřebitele vytvoření produktu JMS :

- **`createDurableConsumer()`**
- **`createSharedDurableConsumer()`**

Režim potvrzení relace

Chcete-li zaručit, že spotřebovaná zpráva byla před odebráním z fronty odběrů IBM MQ plně zpracována, vytvořte relaci JMS pomocí konzoly **`Session`**.Režim CLIENT_POTVRDIT a použít metodu **`message.acknowledge()`** k potvrzení této zprávy a všech ostatních dříve přijatých v této relaci.

Související pojmy

Vývoj klientských aplikací AMQP
Podpora IBM MQ pro rozhraní API AMQP umožňuje administrátorovi systému IBM MQ vytvořit kanál AMQP. Při spuštění tento kanál definuje číslo portu, které přijímá připojení z klientských aplikací AMQP.

V 9.3.3 Odebrání potvrzených zpráv AMQP z fronty v dávkách

Pokud aplikace AMQP používá doručení zprávy QOS_AT_LEAST_ONCE (1), čeká služba AMQP na potvrzení od aplikace, než vyřadí kopii zprávy, kterou uchová poté, co tuto zprávu odešle aplikaci. Z produktu IBM MQ 9.3.3jsou potvrzené zprávy odebírány z fronty v dávkách, nikoli jednotlivě, což vede ke zlepšení výkonu.

Informace o této úloze

V případě zpráv Long Term Support a Continuous Delivery před IBM MQ 9.3.3je každá zpráva odebrána z fronty jednotlivě.

Z produktu IBM MQ 9.3.3můžete použít dvě systémové vlastnosti **`com.ibm.mq.AMQP.BATCHSZ`** a **`com.ibm.mq.AMQP.BATCHINT`** k vyladění zpracování potvrzení v dávkách pro zlepšení výkonu:

com.ibm.mq.AMQP.BATCHSZ
Tento atribut definuje maximální počet potvrzení, která mají být přijata, než služba AMQP odebere zprávy. Číslo může být v rozsahu 1 až 9999. Je-li nastaveno neplatné číslo nebo je-li zadané číslo mimo rozsah, použije se výchozí hodnota 50.

Velikost dávky nemá vliv na způsob přenosu zpráv. Zprávy jsou vždy přenášeny jednotlivě, ale poté jsou odebrány v dávce poté, co služba AMQP obdrží potvrzení. Skutečná velikost dávky může být menší než hodnota určená parametrem **com.ibm.mq.AMQP.BATCHINT**. Například dávka se dokončí, pokud vyprší období nastavené atributem **com.ibm.mq.AMQP.BATCHINT** .

com.ibm.mq.AMQP.BATCHINT
Tento atribut definuje dobu v milisekundách, po kterou služba AMQP uchovává potvrzené zprávy ve frontě. Pokud dávka není plná, je po této době trvání vymazána. Můžete zadat libovolný počet milisekund, od 1 do 999 999 999 999. Výchozí hodnota je 50. Pokud neuvedete hodnotu pro tento atribut, použije se výchozí hodnota 50.

Notes:

1. To, zda služba AMQP čeká na potvrzení před zrušením zprávy, závisí na tom, kterou z následujících dvou kvalit služby aplikace používá pro doručování zpráv:
   - QOS pro QOS_AT_MOST_ONCE (0)

     Pokud aplikace AMQP používá tuto kvalitu služby, nepotvrdí zprávy, takže služba AMQP vyřadí zprávy poté, co je odešle do aplikace, aniž by čekala na potvrzení.
   - QOS_AT_LEAST_ONCE (1)

     Pokud aplikace AMQP používá tuto kvalitu služby, potvrdí zprávy, takže služba AMQP uchová kopii každé zprávy poté, co ji odešle aplikaci, dokud neobdrží potvrzení od aplikace. Pokud se aplikace odpojí nebo ztratí připojení před potvrzením zprávy, je zpráva zpřístupněna ostatním aplikacím. Služba AMQP neodebere zprávu z fronty, dokud nebude potvrzena.
2. MQ Appliance Systémové vlastnosti **com.ibm.mq.AMQP.BATCHSZ** a **com.ibm.mq.AMQP.BATCHINT** nelze v systému IBM MQ Appliancepoužít. V systému IBM MQ Appliancese použije výchozí hodnota 50.

.

Postup

Pomocí systémových vlastností **com.ibm.mq.AMQP.BATCHSZ** a **com.ibm.mq.AMQP.BATCHINT** můžete vyladit zpracování potvrzení v dávkách.

V produktu IBM MQ 9.3.3při vytvoření správce front obsahuje soubor `amqp_java.properties` následující výchozí hodnoty systémových vlastností:

```
-Dcom.ibm.mq.AMQP.BATCHSIZE=50
-Dcom.ibm.mq.AMQP.BATCHINT=50
```

V závislosti na spotřebované rychlosti zpráv můžete vyladit zpracování potvrzení v dávkách pro zlepšení výkonu. Migrovaný správce front nemá tyto vlastnosti v souboru `amqp_java.properties` . Takže pro migrovaného správce front, nebo pokud nejsou vlastnosti nastaveny, použijí se výchozí hodnoty. Tyto vlastnosti můžete přidat, abyste vyladili hodnoty pro optimalizovaný výkon.

Potvrzené zprávy jsou odebrány v dávkách, když je splněna jedna z následujících podmínek:

- Počet potvrzených zpráv dosáhne hodnoty **com.ibm.mq.AMQP.BATCHSZ**.
- Hodnota **com.ibm.mq.AMQP.BATCHINT** je překročena po spuštění dávky.
- Aplikace před splněním dvou předchozích podmínek frontu nebo téma odpojí nebo zavře.

ALW Topologie pro klienty AMQP s produktem IBM MQ

Příklady topologií, které vám pomohou při vývoji klientů AMQP pro práci s produktem IBM MQ.

Související úlohy

Vytváření a používání kanálů AMQP

Zabezpečení klientů AMQP

ALW Klienti produktu AMQP komunikující prostřednictvím produktu IBM MQ

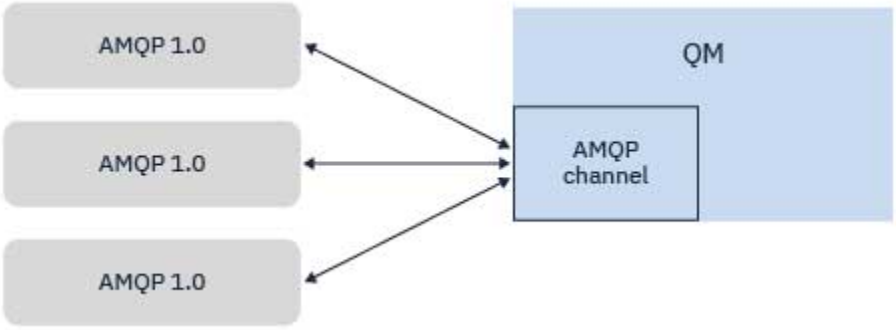
Produkt IBM MQ můžete použít jako poskytovatele systému zpráv pro libovolnou aplikaci, která vyhovuje standardu AMQP 1.0. Ačkoli se kterýkoli klient AMQP 1.0 může připojit ke kanálu AMQP, některé funkce AMQP nejsou podporovány, například transakce nebo vícenásobné relace.

Definováním jednoho nebo více kanálů AMQP se klienti AMQP 1.0 mohou připojit ke správci front a odesílat zprávy do řetězce tématu. Klienti se také mohou přihlásit k odběru vzoru tématu, aby mohli přijímat zprávy, které odpovídají vzoru.

V následujícím scénáři jsou jedinými aplikacemi, které odesílají a přijímají zprávy, aplikace AMQP 1.0 .

Aplikace mohou určit, zda mají být místa určení vytvořená přihlášením k odběru řetězce tématu trvalá, aby v případě, že aplikace dočasně ztratí připojení ke správci front, nebyly zprávy ztraceny.

Aplikace mohou také zvolit, jak dlouho jsou zprávy uchovávány, než jsou vyprázdněny z místa určení.



Související úlohy

Vytváření a používání kanálů AMQP

Zabezpečení klientů AMQP

ALW Klienti produktu AMQP, kteří si vyměňují zprávy s aplikacemi IBM MQ

Definováním a spuštěním kanálu AMQP mohou aplikace AMQP 1.0 publikovat zprávy přijaté existujícími aplikacemi MQ . Všechny zprávy publikované prostřednictvím kanálu AMQP jsou odesílány do témat produktu MQ , nikoli do front MQ . Aplikace MQ , která vytvořila odběr pomocí volání MQSUB API, přijímá zprávy publikované aplikacemi AMQP 1.0 za předpokladu, že řetězec tématu nebo objekt tématu používaný aplikací MQ odpovídá řetězci tématu publikovanému klientem AMQP.

Data, atributy a vlastnosti zprávy AMQP jsou nastaveny ve zprávě MQ přijaté aplikací MQ . Další informace o mapování zpráv AMQP na MQ naleznete v tématu “Mapování polí AMQP na pole IBM MQ (příchozí zprávy)” na stránce 666.

Pokud aplikace MQ vytvořila trvalý odběr, jsou zprávy publikované aplikací AMQP uloženy ve frontě, která daný odběr vrací. Poté jsou zprávy přijímány aplikací MQ , když aplikace pokračuje v odběru. Pokud aplikace AMQP určuje dobu platnosti zprávy a aplikace MQ se během doby platnosti znovu nepřipojí, vyprší platnost zprávy z fronty.

Aplikace ASMP 1.0 mohou také přijímat zprávy publikované existujícími aplikacemi MQ . Zprávy publikované aplikacemi produktu MQ do tématu nebo řetězce tématu MQ jsou přijímány aplikací AMQP 1.0 za předpokladu, že se aplikace přihlásí k odběru se vzorem tématu, který odpovídá publikovanému řetězci tématu.

Pokud aplikace AMQP 1.0 uvádí hodnotu doby platnosti pro odběr a aplikace AMQP se odpojí déle, než je doba platnosti, vyprší platnost odběru ze správce front a všechny zprávy uložené ve frontě odběrů budou ztraceny.

Pole MQMD, vlastnosti zprávy a data aplikace jsou nastaveny ve zprávě AMQP přijaté aplikací AMQP. Další informace o mapování zpráv MQ na AMQP viz “Mapování polí IBM MQ na pole AMQP (odchozí zprávy)” na stránce 665.

Související úlohy

Vytváření a používání kanálů AMQP

Zabezpečení klientů AMQP

ALW Konfigurace klientů AMQP pro přímou interakci s aplikacemi ve frontách IBM MQ

V 9.3.0 Implementace produktu IBM MQ AMQP podporuje publikování/odběr a ukazování na bod. Pro každého klienta AMQP, který nepodporuje point to point, použijte následující kroky k odeslání zpráv do fronty nebo přijetí zpráv z fronty.

Přehled

Předpokládejme například, že aplikace získává zprávy ze vstupní fronty `IN_QUEUE` a vkládá je do výstupní fronty `OUT_QUEUE`. Klienti AMQP mohou vkládat zprávy do adresáře `IN_QUEUE`a získávat zprávy z adresáře `OUT_QUEUE` .

Poznámka: Pro samotnou aplikaci nejsou vyžadovány žádné změny.

Aby mohl vydavatel AMQP vložit zprávu do fronty, musíte vytvořit administrativní odběr pro řetězec tématu, do kterého klient AMQP publikuje, s cílem zamýšlené fronty; viz “Odesílání zpráv do aplikace:” na stránce 676.

Aby mohl odběratel AMQP získat zprávy z fronty, je třeba nahradit frontu s aliasem stejného názvu s cílem objektu tématu reprezentujícím řetězec tématu, k jehož odběru je klient AMQP přihlášen; viz “Získávání zpráv z aplikace:” na stránce 677

Odesílání zpráv do aplikace:

Aplikace již vyzvedává zprávy z produktu `IN_QUEUE` a chcete, aby klient AMQP mohl publikovat zprávy, aby mohly být zpracovány aplikací do této fronty.

Chcete-li tak učinit, vytvořte nový administrativní odběr, kde řetězec tématu, ze kterého tento odběr přijímá zprávy, je řetězec tématu, do kterého klient AMQP publikuje. Cílová fronta tohoto odběru je vstupní fronta pro aplikaci `IN_QUEUE`.

Všechny zprávy publikované do definovaného řetězce tématu pro daný administrativní odběr jsou v tomto případě směrovány do definovaného místa určení `IN_QUEUE`.

Za předpokladu, že klient AMQP publikuje do řetězce tématu `/application/in`, můžete vytvořit administrativní odběr `APP_IN`pomocí následujícího příkazu MQSC:

```
DEF SUB(APP_IN) TOPICSTR('/application/in') DEST('IN_QUEUE')
```

Po definování tohoto objektu jsou všechny zprávy publikované do adresáře `/application/in` směrovány do místa určení `IN_QUEUE`, kde jsou vyzvednuty aplikací stejným způsobem jako všechny ostatní zprávy vkládané do této fronty jinými aplikacemi.

Získávání zpráv z aplikace:

Aplikace vkládá zprávy do produktu `OUT_QUEUE`, kde je mohou vyzvednout a zpracovat jiní klienti.

V tomto případě však chcete, aby byly zprávy doručeny klientovi AMQP, ale klienti AMQP používají pouze publikování/odběr a nemohou zprávy vyzvednout přímo z fronty.

Chcete-li nahradit klienty, kteří dříve přijímali zprávu, odebírajícím klientem AMQP, musíte vytvořit objekt tématu pro řetězec tématu, k jehož odběru je klient AMQP přihlášen, a pro frontu aliasů.

Upozornění: Pokud definujete alias fronty a poté spustíte produkující aplikaci dříve, než budou mít klienti AMQP možnost se přihlásit k odběru, zprávy, které produkující aplikace odešle do "fronty" (nyní téma), budou ztraceny, protože nejsou žádní odběratelé.

Změny popsané v tomto textu nahradí klienty, kteří dříve přijímali zprávy, pouze odebírajícím klientem AMQP. Chcete-li k získání zpráv použít kombinaci AMQP a dalších klientů, je třeba provést rozsáhlejší změny.

Za předpokladu, že se klient AMQP přihlásí k odběru řetězce tématu `/application/out`, můžete definovat objekt tématu APP_OUT pomocí následujícího příkazu MQSC:

```
DEF TOPIC(APP_OUT) TOPICSTR('/application/out')
```

Všechny zprávy doručené do tohoto objektu tématu jsou doručeny klientovi AMQP, který odebírá stejný řetězec tématu.

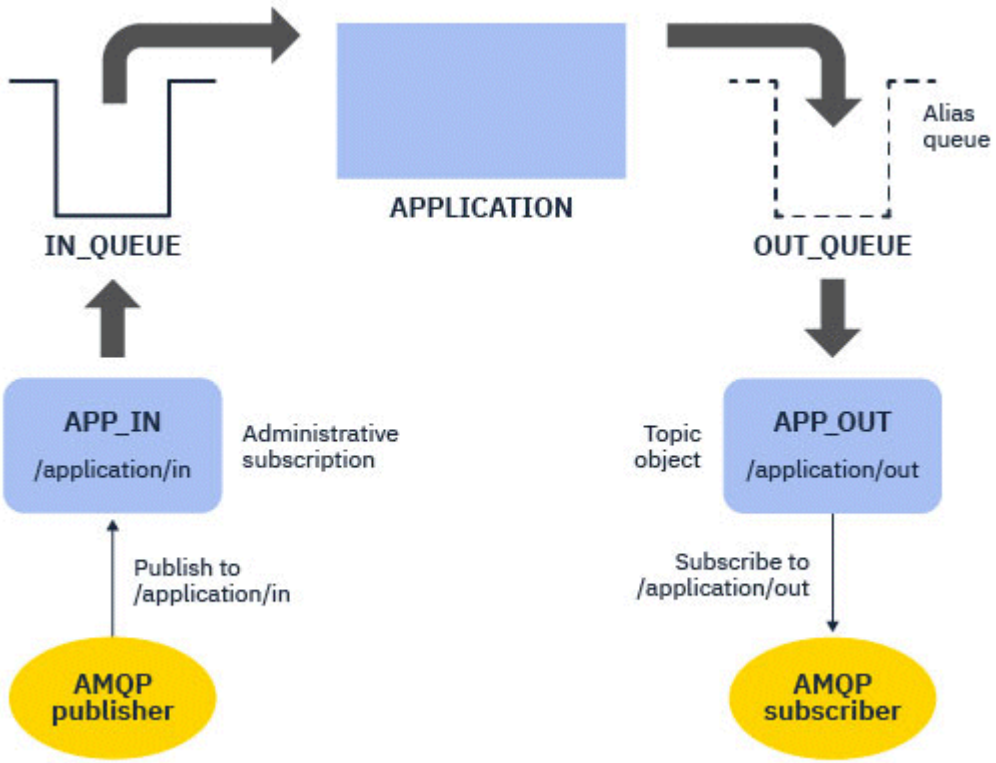
Poté je třeba zajistit, aby zprávy, které aplikace vloží do produktu OUT_QUEUE , byly doručeny do tohoto nového objektu tématu, aby byly odeslány odebírajícím klientovi.

Chcete-li tak učinit, nahraďte existující frontu OUT_QUEUE aliasem fronty se stejným názvem, s typem cíle právě vytvořeného objektu tématu pomocí následujícího příkazu MQSC:

```
DEF QALIAS(OUT_QUEUE) TARGTYPE(TOPIC) TARGET(APP_OUT)
```

Nyní zprávy, které aplikace vložila do OUT_QUEUE , nečekají na vyzvednutí ve frontě; místo toho jsou doručeny do cíle této alias fronty, tj. do nového objektu tématu APP_OUT.

Klient AMQP, který je přihlášen k odběru řetězce tématu reprezentovaného tímto objektem tématu `/application/out`, pak přijme všechny zprávy odeslané do tohoto objektu tématu z alias fronty.



Související úlohy

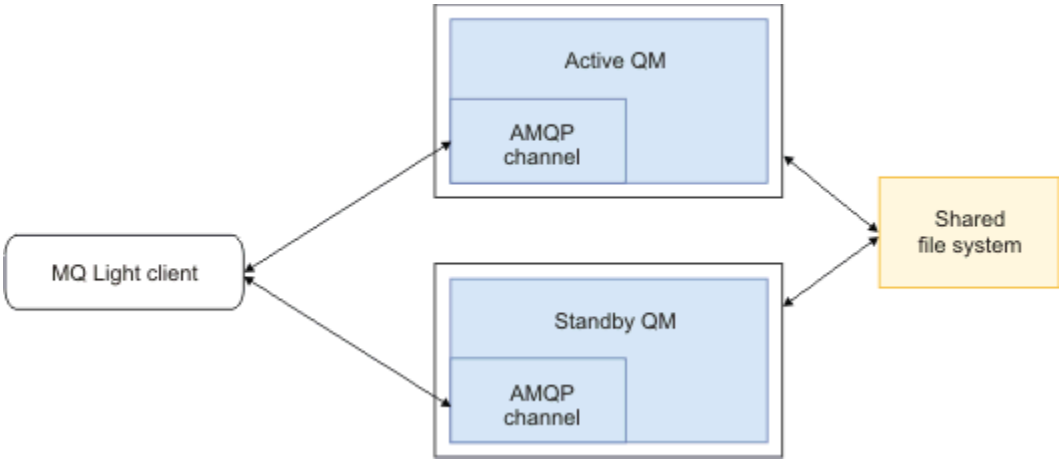
Vytváření a používání kanálů AMQP

Zabezpečení klientů AMQP

ALW Konfigurace klienta AMQP pro vysokou dostupnost

Můžete nakonfigurovat aplikace AMQP 1.0 tak, aby se připojovaly k aktivní instanci správce front s více instancemi IBM MQ a aby se při selhání připojovaly k rezervní instanci správce front s více instancemi ve dvojici vysoké dostupnosti (HA). Chcete-li to provést, nakonfigurujte aplikaci AMQP se dvěma adresami IP a dvojicemi portů.

Rozhraní API klienta AMQP můžete nakonfigurovat pomocí vlastní funkce, která je volána v případě, že klient ztratí připojení k serveru. Funkce se může připojit k alternativní adrese IP, například k pohotovostnímu správci front IBM MQ nebo k původní adrese IP. Pro ostatní klienty AMQP platí, že pokud klient podporuje konfiguraci více koncových bodů připojení, konfigurujte aplikaci se dvěma dvojicemi hostitel-port a pomocí funkcí opětovného připojení, které poskytuje knihovna AMQP, přepněte na správce front v pohotovostním režimu.



Související úlohy

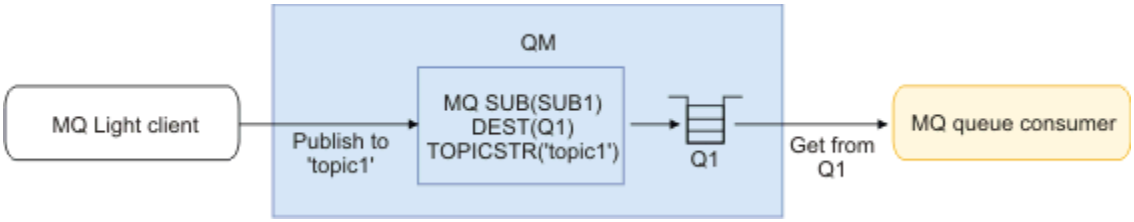
Vytváření a používání kanálů AMQP

Zabezpečení klientů AMQP

ALW Konfigurace publikování/odběru pro klienty AMQP

Klienti produktu AMQP mohou publikovat v tématu s odběrem produktu IBM MQ , který směruje zprávy do fronty IBM MQ načtené existující aplikací. Chcete-li, aby aplikace AMQP 1.0 odesílala zprávy do existující aplikace IBM MQ , která je konfigurována pro čtení z fronty, musíte definovat spravovaný odběr IBM MQ ve správci front.

Nakonfigurujte odběr tak, aby používal vzor tématu, který odpovídá řetězci tématu používanému aplikací AMQP. Nastavte cíl odběru na název fronty, ze které aplikace IBM MQ získává nebo prochází zprávy.



Související úlohy

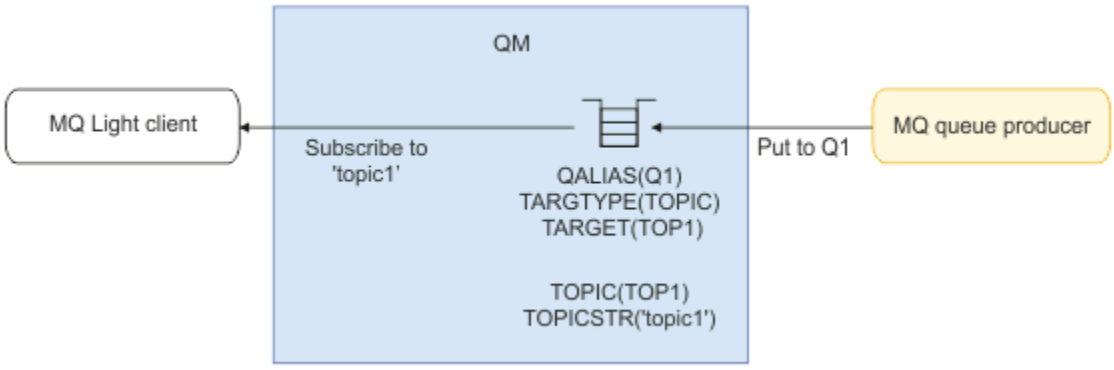
Vytváření a používání kanálů AMQP

Zabezpečení klientů AMQP

ALW Klient AMQP používající alias fronty pro příjem zpráv z aplikace IBM MQ

Klient AMQP se může přihlásit k odběru tématu a přijímat zprávy vkládané do alias fronty aplikací IBM MQ . Chcete-li, aby aplikace AMQP 1.0 přijímala zprávy z existující aplikace IBM MQ , která je konfigurována pro vložení zpráv do fronty, musíte ve správci front definovat alias fronty (QALIAS).

Alias fronty musí mít stejný název jako fronta, kterou aplikace IBM MQ otevře pro vložení. Alias fronty musí určovat základní typ TOPIC a základní objekt objektu tématu IBM MQ , který má řetězec tématu odpovídající vzoru tématu přihlášenému k odběru aplikací AMQP.



Související úlohy

Vytváření a používání kanálů AMQP

Zabezpečení klientů AMQP

ALW Klient AMQP odesílající požadavky na aplikační server a spotřebovávající odpovědi z aplikačního serveru

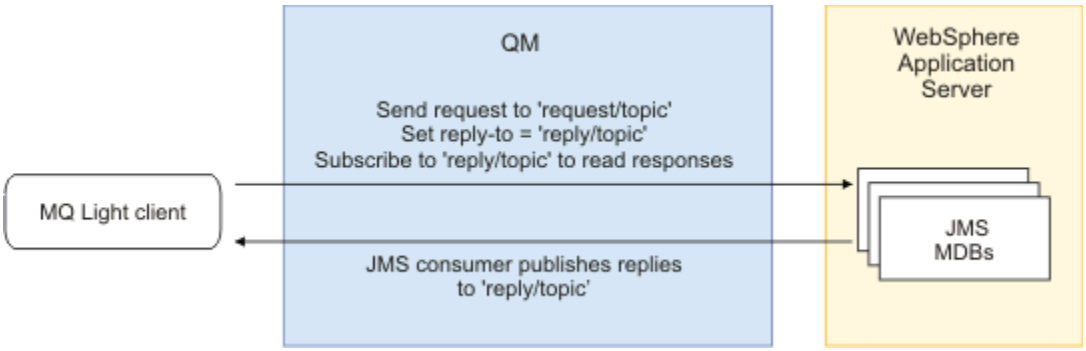
Klient AMQP může odesílat požadavky na objekt typu message-driven bean spuštěný na aplikačním serveru a přijímat odpovědi z tématu odpovědi. IBM MQ podporuje aplikace AMQP 1.0 nastavující téma pro odpověď ve zprávách, které IBM MQ publikuje. Když je zpráva AMQP publikována s nastaveným atributem reply-to, hodnota pole reply-to je nastavena jako vlastnost JMS pro příjem spotřebitelů JMS. Toto nastavení umožňuje spotřebitelům JMS číst téma odpovědi ze zprávy a odeslat zprávu odpovědi zpět klientovi AMQP.

Vlastnost JMS je **JMSReplyTo**. Řetězec odpovědi AMQP musí být jeden z následujících typů:

- Řetězec tématu. Například: `'reply/topic'`
- Adresa AMQP URL ve formátu `amqp://host:port/[topic-string]`. Například `amqp://localhost:5672/reply/topic`

Pokud uvedete adresu AMQP URL jako pole pro odpověď, vše kromě řetězce tématu na konci URL se odebere před nastavením vlastnosti **JMSReplyTo** .

Další informace o mapování z odpovědi AMQP na adresu vlastnosti **JMSReplyTo** viz “Mapování polí AMQP na pole IBM MQ (příchozí zprávy)” na stránce 666



Související úlohy

Vytváření a používání kanálů AMQP

Zabezpečení klientů AMQP

ALW Interoperabilita mezi aplikacemi MQ Light a Apache Qpid JMS

Aplikace MQ Light a Apache Qpid JMS pracují podobným způsobem a při přihlášení k odběru tématu vytvářejí odběry IBM MQ , které se řídí stejnou konvencí pojmenování.

Soukromý, nesdílený odběr

Název odběru IBM MQ vytvořeného aplikací je `:private:<clientid>:<topicstring>`.

Aplikace používající jiné ID klienta nemůže přistupovat k odběrům vytvořeným jinými aplikacemi, protože název odběru je automaticky generován a obsahuje ID klienta AMQP.

Aplikace Apache Qpid JMS i MQ Light používají tuto konvenci pojmenování pro soukromé odběry.

Globálně sdílené odběry

Název globálně sdíleného odběru IBM MQ vytvořeného klientem AMQP je `:share:<sharename>:<topicstring>`.

Pokud několik aplikací s různými ID klienta AMQP určuje stejný název sdílení a řetězec tématu, sdílejí jeden odběr a mohou spolupracovat na zpracování zpráv pro daný odběr. Tento vzor můžete použít, chcete-li změnit počet pracovních aplikací vyprazdňujících zprávy z odběru.

Aplikace Apache Qpid JMS i MQ Light používají tuto konvenci pojmenování pro globálně sdílené odběry. V případě Apache Qpid JMSto vyžaduje, aby připojení JMS nemělo uvedeno ID klienta.

Knihovna Apache Qpid JMS generuje ID klienta AMQP automaticky, ale toto ID klienta se nepoužívá pro účely pojmenování odběru IBM MQ .

Poznámka: Globálně sdílené odběry jsou stále vyhrazeny pro jednotlivé správce front.

Soukromé sdílené odběry

Název soukromě sdíleného odběru IBM MQ vytvořeného klientem AMQP je `:privateshare:<clientid>:<sharename>:<topicstring>`.

Pokud několik podprocesů z jedné aplikace Apache Qpid JMS používá stejný název sdílení a řetězec tématu a ID klienta bylo nakonfigurováno v připojení JMS , tyto podprocesy sdílejí stejný objekt odběru IBM MQ .

Jiná připojení Apache Qpid JMS však nemohou sdílet odběr, protože musí používat jiné ID klienta.

Klienti systému MQ Light nepodporují koncept soukromých sdílených odběrů a nemohou přijímat zprávy ze soukromých sdílených odběrů vytvořených aplikací Apache Qpid JMS .

IBM MQ JMS odběry

Odběry produktu IBM MQ JMS používají jiné schéma pojmenování než kanály AMQP. Není možné, aby aplikace MQ Light nebo Apache Qpid JMS sdílely odběry s aplikacemi IBM MQ JMS .

Související pojmy

Vývoj klientských aplikací AMQP

Podpora IBM MQ pro rozhraní API AMQP umožňuje administrátorovi systému IBM MQ vytvořit kanál AMQP. Při spuštění tento kanál definuje číslo portu, které přijímá připojení z klientských aplikací AMQP.

ALW IBM MQ Vlastnosti ovládacího prvku listeneru AMQP

Chcete-li dosáhnout lepšího výkonu v aplikaci s podporou podprocesů, můžete vyladit počet pracovních podprocesů, které by měla služba AMQP použít, konfigurací vlastnosti v souboru vlastností AMQP.

Vlastnosti služby modulu listener AMQP můžete konfigurovat v následujících souborech vlastností:

- Windows Na systémech Windows : `amqp_win.properties` .
- Linux AIX Na systémech AIX and Linux : `amqp_unix.properties` .

Vlastnosti, které můžete konfigurovat, jsou následující:

Tabulka 105. Vlastnosti služby modulu listener AMQP

Vlastnost	Popis
com.ibm.mq.MQXR.Workers	Počet pracovních podprocesů serveru, které služba modulu listener AMQP vytvoří. Není-li tato hodnota uvedena, standardně se rovná počtu logických procesorů v systému.
MQIBindType	Typ vazby pro službu AMQP: FASTPATH, SHAREDnebo ISOLATED. Výchozí hodnota je FASTPATH.

Služba modulu listener AMQP vyvažuje pracovní zátěž připojení klienta v rámci počtu pracovních podprocesů. Počet pracovních podprocesů, které by měla služba AMQP použít, lze uvést pomocí vlastnosti **com.ibm.mq.MQXR.Workers** .

Administrátor správce front IBM MQ může vyladit počet pracovních podprocesů pro lepší výkon v aplikaci s podporou podprocesů. Obvykle je nejlepšího výkonu dosaženo, když se počet pracovních podprocesů

shoduje s počtem logických procesorů v systému. To však nemusí být vždy případ určitých konfigurací počítače a charakteristik zatížení klienta, takže prvek ladění může být nezbytný k nalezení optimální hodnoty pro počet pracovních podprocesů.

Před vyladěním se ujistěte, že jste důkladně porozuměli povaze klientských aplikací a jejich pracovní zátěži. Měření výkonu aplikace s různými počty podprocesů a benchmarkingem by mělo pomoci určit optimální hodnotu pro počet pracovních podprocesů.

Poznámka: MQ Appliance Tyto vlastnosti nelze použít v systému IBM MQ Appliance. Výchozí hodnoty jsou použity v systému IBM MQ Appliance.

Vývoj aplikací REST pomocí produktu IBM MQ

Můžete vyvíjet aplikace REST pro odesílání a přijímání zpráv. Produkt IBM MQ podporuje různá rozhraní REST API v závislosti na platformě a schopnostech.

Následující volby jsou IBM MQ podporované volby, ze kterých si můžete vybrat, chcete-li odesílat zprávy do a přijímat zprávy z produktu IBM MQ:

- IBM MQ messaging REST API
- IBM z/OS Connect EE
- IBM Integration Bus
- DataPower

IBM MQ messaging REST API

Pomocí konzoly messaging REST API můžete odesílat, přijímat a procházet zprávy produktu IBM MQ ve formátu prostého textu. Volba messaging REST API je standardně povolena.

Podpora je poskytována pro řadu různých záhlaví HTTP , která lze použít k nastavení společných vlastností zprávy.

Produkt messaging REST API je plně integrován se zabezpečením IBM MQ . Chcete-li používat produkt messaging REST API, musí být uživatelé ověřeni na serveru mqweb a musí být členem role MQWebUser .

Další informace uvádí téma “Zasílání zpráv pomocí konzoly REST API” na stránce 683. Viz také Výukový program: Začínáme s produktem IBM MQ messaging REST API na webu IBM Developer, který zahrnuje příklady Go a Node.js pro použití rozhraní API REST systému zpráv.

IBM z/OS Připojit EE

IBM z/OS Connect EE je produkt z/OS , který vám umožňuje sestavit rozhraní REST API nad existujícími aktivy z/OS , jako jsou CICS nebo IMS transakce, a IBM MQ fronty a témata. Existující aktivum z/OS je uživateli skryto. To vám umožňuje povolit vaše aktiva REST, aniž byste je měnili nebo jakékoli existující aplikace, které je používají.

Produkt IBM z/OS Connect EE poskytuje automatickou transformaci dat pro překlad mezi daty JSON používanými rozhraními REST API a tradičnějšími jazykovými strukturami, například COBOL, očekávanými mnoha aplikacemi sálového počítače.

Sadu nástrojů Eclipse based IBM z/OS Connect EE API Toolkit lze použít k sestavení komplexního rozhraní RESTful API s využitím parametrů dotazu a segmentů cesty URL , které manipulují s formátem JSON při průchodu běhovým prostředím produktu IBM z/OS Connect EE.

Produkt IBM z/OS Connect EE lze použít k vystavení IBM MQ front a témat jako rozhraní API RESTful prostřednictvím poskytovatele služeb IBM MQ . Jsou podporovány dva různé typy služeb:

- Jednosměrné služby: poskytují rozhraní REST API, které umožňuje provedení jedné operace IBM MQ na frontě nebo tématu. V závislosti na přesné konfiguraci může požadavek HTTP vést k odeslání zprávy do fronty nebo k publikování do tématu; nebo může požadavek HTTP vést k destruktivnímu přijetí zprávy z fronty.

- Obousměrné služby: poskytují rozhraní REST API nad dvojicí front používaných aplikací typu požadavek-odezva back-endového systému. Volající vydají požadavek HTTP na obousměrnou službu. Informační obsah požadavku HTTP je převeden z formátu JSON na tradiční strukturu jazyka a vložen do fronty požadavků, kde je zpracován back-endovou aplikací a odezvou umístěnou do fronty odpovědí. Tato odpověď je načtena službou, převedena ze struktury tradičního jazyka do formátu JSON a odeslána zpět volajícímu jako tělo odezvy POST.

Další informace o produktu IBM z/OS Connect EE viz z/OS Connect EE.

Další informace o poskytovateli služeb IBM MQ naleznete v tématu Použití IBM MQ poskytovatele služeb.

IBM Integration Bus

IBM Integration Bus je IBM, kterou lze použít k propojení aplikací a systémů bez ohledu na formáty zpráv a protokoly, které podporují.

Produkt IBM Integration Bus vždy podporoval produkt IBM MQ a poskytuje uzly *HTTPInput* a *HTTPRequest* , které lze použít k vytvoření rozhraní RESTful na serveru IBM MQa mnoha dalších systémech, jako jsou databáze.

Produkt IBM Integration Bus lze použít k mnohem více, než k poskytnutí jednoduchého rozhraní REST v produktu IBM MQ. Jeho schopnosti lze použít k poskytnutí rozšířené manipulace s informačním obsahem, obohacení informačního obsahu a mnoha dalších vylepšení jako součást produktu REST API.

Další informace viz ukázka technologie , která vystavuje rozhraní JSON přes rozhraní REST nad aplikací IBM MQ , která očekává informační obsah XML.

DataPower

Brána DataPower je jediná vícekanálová brána, která pomáhá zajistit zabezpečení, řízení, integraci a optimalizovaný přístup k řadě systémů, včetně systému IBM MQ. Dodává se v hardwarových i virtuálních formových faktorech.

Jedna ze služeb, které produkt DataPower poskytuje, je víceprotokolová brána, která může přijmout vstup v jednom protokolu a generovat výstup v jiném protokolu. Konkrétně produkt DataPower lze nakonfigurovat tak, aby přijímal data HTTP(S) a směroval je do produktu IBM MQ přes připojení klienta, které lze použít k sestavení rozhraní REST v horní části produktu IBM MQ. Další služby DataPower , jako např. transformace, lze také použít k rozšíření rozhraní REST.

Další informace viz Multi-Protocol Gateway.

Zasílání zpráv pomocí konzoly REST API

Pomocí konzoly messaging REST API můžete provádět jednoduché dvoubodové zprávy a publikovat zprávy. Můžete publikovat zprávy do tématu, odesílat zprávy do fronty, procházet zprávy ve frontě a destruktivně získávat zprávy z fronty. Informace jsou odesílány do souboru messaging REST API a přijímány z něj ve formátu prostého textu.

Než začnete

Poznámka:

- Volba messaging REST API je standardně povolena. Funkci messaging REST API můžete zakázat, chcete-li zabránit všem zprávám. Další informace o povolení nebo zakázání konzoly messaging REST APInaleznete v tématu Konfigurace konzoly messaging REST API.
- Produkt messaging REST API je integrován se zabezpečením IBM MQ . Chcete-li používat produkt messaging REST API, musí být uživatelé ověřeni na serveru mqweb a musí být členem role MQWebUser . Uživatel musí být také autorizován pro přístup k uvedené frontě nebo tématu. Další informace o zabezpečení REST APInaleznete v tématu Zabezpečení produktů IBM MQ Console a REST API.

- Používáte-li systém Advanced Message Security (AMS) s produktem messaging REST API, mějte na paměti, že všechny zprávy jsou šifrovány pomocí kontextu serveru mqweb, nikoli kontextu uživatele, který zprávu zveřejňuje.
- Při příjmu nebo procházení zprávy jsou podporovány pouze zprávy ve formátu IBM MQ `MQSTR` nebo `JMS TextMessage` . Následně jsou všechny zprávy destruktivně přijímány v synchronizačním bodu a všechny neošetřené zprávy jsou ponechány ve frontě. Frontu IBM MQ lze konfigurovat tak, aby se tyto nezpracovatelné zprávy přesunovaly do alternativního místa určení. Další informace uvádí téma “Zpracování nezpracovatelných zpráv v adresáři IBM MQ classes for JMS” na stránce 225.
- Produkt messaging REST API vám neposkytuje pouze jednou a pouze jednou doručení zpráv s podporou transakcí. Je-li vydán požadavek HTTP POST a připojení se nezdaří před přijetím odpovědi HTTP klientem, klient nemůže okamžitě zjistit, zda byla zpráva odeslána do určené fronty nebo publikována do určeného tématu. Pokud je vydán příkaz HTTP DELETE a připojení selže před přijetím odpovědi HTTP klientem, pak mohla být zpráva destruktivně získána z fronty a ztracena, protože neexistuje žádný způsob, jak vrátit destruktivní získání zpět.
- V 9.3.0 Z operačního systému IBM MQ 9.3.0již operace HTTP POST neodebírá nové řádky v příchozích řetězcích. Aplikace REST, které používají starší verze, by neměly používat nové řádky ve zprávách, které jsou odesílány nebo publikovány pomocí rozhraní REST API, protože budou ztraceny.

Procedura

- “Začínáme s produktem messaging REST API” na stránce 684
- “Použití produktu messaging REST API” na stránce 687
- REST API ošetření chyb
- REST API zjišťování
- REST API národní jazyková podpora

Související odkazy

Odkaz na systém zpráv REST API

Související informace

Výukový program: Začínáme se systémem zpráv IBM MQ REST API

Začínáme s produktem messaging REST API

Začněte rychle s produktem messaging REST API a vyzkoušejte několik příkladů příkazů pomocí příkazu cURL.

Než začnete

Chcete-li začít používat produkt messaging REST API, příklady v této úloze mají následující požadavky:

- Příklady používají cURL k odeslání požadavků REST pro vložení a získání zpráv z fronty. Chcete-li tedy provést tuto úlohu, musíte ve svém systému nainstalovat cURL .
- Příklady používají správce front `QM1`. Buď vytvořte správce front se stejným názvem, nebo nahraďte existujícího správce front v systému. Správce front musí být na stejném počítači jako server mqweb.
- Chcete-li dokončit tuto úlohu, musíte být uživatelem s určitými oprávněními, abyste mohli použít příkaz **dspmqweb**:
  - z/OS V systému z/OS musíte mít oprávnění ke spuštění příkazu **dspmqweb** a k zápisu do souboru `mqwebuser.xml`.
  - Multi U všech ostatních operačních systémů musíte být privilegovaný uživatel.

  IBM i V systému IBM iby měly být příkazy spuštěny v QSHELL.

Postup

1. Ujistěte se, že je server mqweb nakonfigurován pro messaging REST API:
   - Ujistěte se, že jste nakonfigurovali server mqweb pro použití administrative REST API, administrative REST API pro MFT, messaging REST APInebo IBM MQ Console.Další informace o konfiguraci serveru mqweb se základním registrem naleznete v tématu Základní konfigurace serveru mqweb.
   - Pokud je server mqweb již nakonfigurován, ujistěte se, že jste přidali příslušné uživatele pro povolení systému zpráv v kroku 5 Základní konfigurace pro server mqweb.
     - Je-li parametr **mqRestMessagingAdoptWebUserContext** v konfiguraci serveru mqweb nastaven na hodnotu `true` , uživatelé produktu messaging REST API musí být členem role `MQWebUser` . Role `MQWebAdmin` a `MQWebAdminRO` nelze použít pro messaging REST API. Uživatelé musí být také autorizováni pro přístup k frontám a tématům, která se používají pro zasílání zpráv prostřednictvím OAM nebo RACF.
     - Je-li parametr **mqRestMessagingAdoptWebUserContext** v konfiguraci serveru mqweb nastaven na hodnotu `false` , musí být ID uživatele, které se používá ke spuštění serveru mqweb, autorizováno pro přístup k frontám používaným pro zasílání zpráv prostřednictvím OAM nebo RACF.
2. z/OS
   V systému z/OSnastavte proměnnou prostředí WLP_USER_DIR tak, abyste mohli použít příkaz **dspmqweb** . Nastavte proměnnou tak, aby ukazovala na konfiguraci serveru mqweb, zadáním následujícího příkazu:

   ```
   export WLP_USER_DIR=WLP_user_directory
   ```

   , kde *WLP_user_directory* je název adresáře, který se předává do `crtmqweb`. Například:

   ```
   export WLP_USER_DIR=/var/mqm/web/installation1
   ```

   Další informace viz Vytvoření serveru mqweb.
3. Určete adresu URL REST API zadáním následujícího příkazu:

   ```
   dspmqweb status
   ```

   Příklady v následujících krocích předpokládají, že vaše REST API URL je výchozí URL `https://localhost:9443/ibmmq/rest/v2/`. Pokud se vaše adresa URL liší od výchozí adresy, nahraďte ji následujícími kroky.
4. Vytvořte frontu `MSGQ`ve správci front `QM1`. Tato fronta se používá pro zasílání zpráv. Postupujte některým z následujících způsobů:
   - Použijte požadavek POST na prostředek `mqsc` serveru administrative REST API, který se ověřuje jako uživatel `mqadmin` :

     ```
     curl -k https://localhost:9443/ibmmq/rest/v2/admin/action/qmgr/QM1/mqsc -X POST -u
     mqadmin:mqadmin -H "ibm-mq-rest-csrf-token: value" -H "Content-Type: application/
     json" --data "{\"type\": \"runCommandJSON\",\"command\": \"define\", \"qualifier\":
     \"qlocal\",\"name\": \"MSGQ\"}"
     ```

   - Použít příkazy MQSC:

     z/OS V systému z/OSpoužijte místo příkazu **runmqsc** zdroj 2CR . Další informace naleznete v tématu Zdroje, z nichž můžete zadat příkazy MQSC a PCF v systému IBM MQ for z/OS.

     a. Spusťte **runmqsc** pro správce front zadáním následujícího příkazu:

     ```
     runmqsc QM1
     ```

b. Pomocí příkazu **DEFINE QLOCAL** MQSC vytvořte frontu:

```
DEFINE QLOCAL(MSGQ)
```

c. Ukončete **runmqsc** zadáním následujícího příkazu:

```
end
```

5. Udělte oprávnění uživateli, kterého jste přidali do souboru `mqwebuser.xml` v kroku 5 Základní konfigurace pro server mqweb pro přístup k frontě `MSGQ`. Nahraďte svého uživatele, kde se používá `myuser`:

- z/OS V systému z/OS:

  a. Udělte uživateli přístup k frontě:

  ```
  RDEFINE MQQUEUE hlq.MSGQ UACC(NONE)
  PERMIT hlq.MSGQ CLASS(MQQUEUE) ID(MYUSER) ACCESS(UPDATE)
  ```

  b. Udělte ID uživatele spuštěné úlohy mqweb přístup k nastavení všech kontextů ve frontě:

  ```
  RDEFINE MQADMIN hlq.CONTEXT.MSGQ UACC(NONE)
  PERMIT hlq.CONTEXT.MSGQ CLASS(MQADMIN) ID(mqwebStartedTaskID) ACCESS(CONTROL)
  ```

- Multi Ve všech ostatních operačních systémech, pokud je váš uživatel ve skupině mqm, je oprávnění již uděleno. Jinak zadejte následující příkazy:

  a. Spusťte **runmqsc** pro správce front zadáním následujícího příkazu:

  ```
  runmqsc QM1
  ```

  b. Pomocí příkazu **SET AUTHREC** MQSC můžete uživateli poskytnout oprávnění k procházení, dotazování, získávání a vložení do fronty:

  ```
  SET AUTHREC PROFILE(MSGQ) OBJTYPE(QUEUE) +
  PRINCIPAL(myuser) AUTHADD(BROWSE, INQ, GET, PUT)
  ```

  c. Ukončete **runmqsc** zadáním následujícího příkazu:

  ```
  end
  ```

6. Vložte zprávu s obsahem `Hello World!` do fronty `MSGQ` ve správci front `QM1`pomocí požadavku POST na prostředek `message` . Nahraďte ID uživatele a heslo z `mqwebuser.xml` pro `myuser` a `mypassword`:

   Používá se základní ověření a v požadavku REST cURL je nastaveno záhlaví `ibm-mq-rest-csrf-token` HTTP s libovolnou hodnotou. Toto další záhlaví je vyžadováno pro požadavky POST, PATCH a DELETE.

   ```
   curl -k https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/MSGQ/message -X
   POST -u myuser:mypassword -H "ibm-mq-rest-csrf-token: value" -H "Content-Type: text/
   plain;charset=utf-8" --data "Hello World!"
   ```

7. Destruktivně získejte zprávu z fronty `Hello World!` ve frontě `MSGQ` ve správci front `QM1`pomocí požadavku DELETE na prostředek `message` . Nahraďte ID uživatele a heslo z `mqwebuser.xml` pro `myuser` a `mypassword`:

   ```
   curl -k https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/MSGQ/message -X DELETE
   -u myuser:mypassword -H "ibm-mq-rest-csrf-token: value"
   ```

   Vrátí se zpráva `Hello World!` .

Jak pokračovat dále

- Příklady používají základní ověření k zabezpečení požadavku. Místo toho můžete použít ověření založené na tokenech nebo ověření na základě klienta. Další informace viz Použití ověření klientského certifikátu pomocí REST API a IBM MQ Console a Použití ověření pomocí tokenů pomocí REST API.
- Další informace o použití konzoly messaging REST API a vytváření adres URL s parametry dotazu: “Použití produktu messaging REST API” na stránce 687.
- Při použití konzoly messaging REST APIjsou připojení ke správci front uložena do fondu pro optimalizaci výkonu. Můžete nakonfigurovat maximální velikost fondu a akci, která se provede, když se všechna připojení ve fondu používají: Konfigurace serveru messaging REST API.
- Procházením referenčních informací vyhledejte dostupné prostředky messaging REST API a všechny dostupné parametry dotazu: messaging REST API odkaz.
- Zjistěte administrative REST API, rozhraní RESTful pro IBM MQ administraci: Administrace pomocí REST API.
- Zjistěte IBM MQ Console, grafické uživatelské rozhraní založené na prohlížeči: Administrace pomocí konzoly IBM MQ Console.

Použití produktu messaging REST API

Při použití messaging REST APIvyvoláte metody HTTP na adresách URL pro odesílání a příjem IBM MQ zpráv. Metoda HTTP , například POST, představuje typ akce, která se má provést na objektu reprezentovaném URL. Další informace o akci mohou být zakódovány v parametrech dotazu. Informace o výsledku provedení akce mohou být vráceny jako tělo odpovědi HTTP .

Než začnete

Před použitím konzoly messaging REST APIzvažte tyto skutečnosti:

- Chcete-li používat soubor messaging REST API, musíte se ověřit na serveru mqweb. Můžete provést ověření pomocí základního ověření HTTP , ověření pomocí certifikátu klienta nebo ověření pomocí tokenu. Další informace o použití těchto metod ověřování naleznete v tématu IBM MQ Console a REST API zabezpečení.
- REST API rozlišuje velikost písmen. Například HTTP POST na následující URL způsobí chybu, pokud se správce front nazývá `qmgr1`.

```
/ibmmq/rest/v2/messaging/qmgr/QMGR1/queue/Q1/message
```

- V 9.3.3 Pokud se připojujete ke vzdálenému správci front pomocí konzoly messaging REST API, musíte místo názvu správce front použít jedinečný název pro připojení správce front.
- Ne všechny znaky, které lze použít v názvech objektů IBM MQ , mohou být přímo zakódovány v URL. Chcete-li tyto znaky správně kódovat, musíte použít příslušné kódování URL :
  - Dopředné lomítko musí být zakódováno jako %2F.
  - Znak procenta musí být zakódován jako %25.
  - Tečka musí být zakódována jako %2E.
  - Otazník musí být zakódován jako %3F.
- Při příjmu nebo procházení zprávy jsou podporovány pouze zprávy ve formátu IBM MQ `MQSTR` a `JMS TextMessage` . Následně jsou všechny zprávy destruktivně přijímány v synchronizačním bodu a všechny neošetřené zprávy jsou ponechány ve frontě. Frontu IBM MQ lze konfigurovat tak, aby se tyto nezpracovatelné zprávy přesunovaly do alternativního místa určení. Další informace uvádí téma “Zpracování nezpracovatelných zpráv v adresáři IBM MQ classes for JMS” na stránce 225.

Informace o této úloze

Když použijete REST API k provedení akce systému zpráv na objektu fronty IBM MQ , musíte nejprve vytvořit URL , která bude tento objekt reprezentovat. Každá URL začíná předponou, která popisuje, na

který název hostitele a port se má požadavek odeslat. Zbytek URL popisuje konkrétní objekt nebo směrování na tento objekt, známý jako prostředek.

Akce systému zpráv, která má být provedena na prostředku, definuje, zda adresa URL potřebuje parametry dotazu, či nikoli. Také definuje použitou metodu HTTP a to, zda jsou na adresu URLodesílány nebo z ní vracené další informace. Další informace mohou tvořit část požadavku HTTP nebo mohou být vráceny jako součást odezvy HTTP .

Po vytvoření URLmůžete odeslat požadavek HTTP na adresu IBM MQ. Požadavek můžete odeslat pomocí implementace HTTP , která je vestavěna do vámi zvoleného programovacího jazyka. Požadavek můžete také odeslat pomocí nástrojů příkazového řádku, jako např. cURL, nebo pomocí webového prohlížeče nebo doplňku webového prohlížeče.

Důležité: Musíte provést minimálně kroky “1.a” na stránce 688 a “1.b” na stránce 688.

Postup

1. Vytvořte URL:
   a) Určete předponu URL zadáním následujícího příkazu:

   ```
   dspmqweb status
   ```

   URL , kterou chcete použít, obsahuje frázi `/ibmmq/rest/` .
   b) Přidejte frontu a přidružené prostředky správce front, které se mají použít pro systém zpráv, do cesty URL .

   V odkazu na systém zpráv lze segmenty proměnných identifikovat v URL složenými závorkami, které jej obklopují *{ }*. Další informace viz /messaging/qmgr/{qmgrName}/queue/{queueName}/message.

   Chcete-li například interaktivně spolupracovat s frontou *Q1* přidruženou ke správci front *QM1*, přidejte `/qmgr` a `/queue` k předponě URL a vytvořte následující URL:

   ```
   https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/message
   ```

   Tip: V 9.3.3 Pokud je správce front vzdáleným správcem front, musíte místo názvu správce front použít jedinečný název správce front. Před použitím s produktem messaging REST APImusí být konfigurován vzdálený správce front. Další informace viz téma “Nastavení vzdáleného správce front pro použití s produktem messaging REST API” na stránce 688.
   c) Volitelné: Přidejte volitelný parametr dotazu do URL.

   Přidat otazník,?, parametr dotazu, rovnítko = a hodnotu URL.

   Chcete-li například čekat maximálně 30 sekund na zpřístupnění další zprávy, vytvořte následující URL:

   ```
   https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/message?wait=30000
   ```

   d) Volitelné: Přidejte další volitelné parametry dotazu do URL.

   Přidejte znak ampersand & do URLa poté zopakujte krok 1c.
2. Vyvolejte příslušnou metodu HTTP na URL. Uveďte libovolný volitelný informační obsah zprávy a poskytněte odpovídající pověření zabezpečení pro ověření. Příklad:
   - Použijte implementaci HTTP/REST zvoleného programovacího jazyka.
   - Použijte nástroj, jako např. doplněk prohlížeče klienta REST nebo cURL.

V 9.3.3 Nastavení vzdáleného správce front pro použití s produktem messaging REST API

Pomocí konzoly messaging REST API se můžete připojit ke vzdáleným správcům front pro systém zpráv. Před připojením ke vzdálenému správci front je třeba nastavit konfiguraci vzdáleného správce front.

Poté se můžete připojit ke vzdálenému správci front pomocí jedinečného názvu, který je definován v informacích o konfiguraci.

Než začnete

- Ujistěte se, že jste nakonfigurovali server mqweb pro použití administrative REST API, administrative REST API pro MFT, messaging REST APInebo IBM MQ Console.Další informace o konfiguraci serveru mqweb se základním registrem naleznete v tématu Základní konfigurace serveru mqweb.
- Pokud je server mqweb již nakonfigurován, ujistěte se, že jste přidali příslušné uživatele pro povolení systému zpráv v kroku 5 Základní konfigurace pro server mqweb. Uživatelé produktu messaging REST API musí být členem role `MQWebUser` . Role `MQWebAdmin` a `MQWebAdminRO` nelze použít pro messaging REST API.
  - Pokud je parametr **`mqRestMessagingAdoptWebUserContext`** v konfiguraci serveru mqweb nastaven na hodnotu `true` , musí být uživatelé v roli `MQWebUser` autorizováni pro přístup k frontám a tématům, která se používají pro zasílání zpráv prostřednictvím OAM nebo RACF.
  - Je-li parametr **`mqRestMessagingAdoptWebUserContext`** v konfiguraci serveru mqweb nastaven na hodnotu `false` , musí být ID uživatele, které se používá ke spuštění serveru mqweb, autorizováno pro přístup k frontám a tématům, která se používají pro zasílání zpráv prostřednictvím OAM nebo RACF.
- Ujistěte se, že je agent messaging REST API konfigurován pro připojení ke vzdáleným správcům front. Další informace naleznete v tématu Konfigurace režimu připojení pro server messaging REST API.

Informace o této úloze

Pomocí konzoly messaging REST APIse můžete připojit ke vzdáleným správcům front. Vzdálený správce front může být správcem front v jiném systému, správcem front v jiné instalaci nebo správcem front ve stejné instalaci jako server mqweb.

Chcete-li se připojit ke vzdálenému správci front, musíte provést následující kroky konfigurace:

- Konfigurace kanálu připojení serveru a modulu listener.
- Udělte příslušnému uživateli oprávnění pro přístup ke správci front.
- Vytvořte soubor CCDT, který obsahuje informace o připojení pro správce front.
- Přidejte informace o připojení do souboru messaging REST API pomocí příkazu **`setmqweb remote`** .

Poté můžete použít vzdáleného správce front zadáním jedinečného názvu v prostředku URL namísto názvu správce front.

Můžete také konfigurovat vzdálené správce front jako součást skupiny správců front. Další informace viz téma “Nastavení skupiny správců front pro použití s rozhraním REST API systému zpráv” na stránce 692.

Postup

1. Ve vzdáleném správci front vytvořte kanál připojení serveru, který umožní vzdálená připojení ke správci front. Kanály připojení serveru můžete vytvořit pomocí příkazu **`DEFINE CHANNEL`** MQSC na příkazovém řádku.
   Chcete-li například vytvořit kanál připojení serveru `QM1.SVRCONN` pro správce front QM1, zadejte následující příkaz:

   ```
   DEFINE CHANNEL(QM1.SVRCONN) CHLTYPE(SVRCONN) TRPTYPE(TCP)
   ```

   Další informace o produktu **`DEFINE CHANNEL`** a dostupných volbách viz DEFINE CHANNEL.
2. Ujistěte se, že příslušný uživatel je autorizován pro přístup ke správci front. Tento uživatel musí být také autorizován pro přístup ke všem frontám nebo tématům, které používáte pro zasílání zpráv. Uživatel potřebuje oprávnění `connect`, `inquire`, `alternate usera set context` ke správci front. V systému UNIX, Linux, and Windows použijte řídicí příkaz **`setmqaut`** na příkazovém řádku. V systému z/OSdefinujte profily RACF , které poskytnou autorizovanému uživateli přístup ke správci front.

Chcete-li například v systému UNIX, Linux, and Windowsautorizovat uživatele `exampleUser`pro přístup ke správci front QM1, zadejte následující příkaz:

```
setmqaut -m QM1 -t qmgr -p exampleUser +connect +inq +altusr +setall
```

Další informace o tom, který uživatel musí být autorizován, viz "Určení činitele zabezpečení používaného produktem messaging REST API" na stránce 694.

3. ALW

Pokud ve vzdáleném správci front neexistuje žádný modul listener, vytvořte modul listener pro příjem příchozích síťových připojení pomocí příkazu **DEFINE LISTENER** MQSC na příkazovém řádku. Chcete-li například vytvořit modul listener `REMOTE.LISTENER` na portu 1414 pro vzdáleného správce front QM1, zadejte následující příkaz:

```
runmqsc QM1
DEFINE LISTENER(REMOTE.LISTENER) TRPTYPE(TCP) PORT(1414)
end
```

4. Ujistěte se, že je modul listener spuštěn pomocí příkazu **START LISTENER** MQSC na příkazovém řádku:

ALW Chcete-li například v systému AIX, Linux, and Windows spustit modul listener `REMOTE.LISTENER` pro správce front QM1, zadejte následující příkaz:

```
runmqsc QM1
START LISTENER(REMOTE.LISTENER)
end
```

z/OS Chcete-li například v systému z/OSspustit modul listener, zadejte následující příkaz:

```
runmqsc QM1
START LISTENER TRPTYPE(TCP) PORT(1414)
end
```

Před spuštěním modulu listener v systému z/OSmusí být spuštěn adresní prostor inicializátoru kanálu.

5. V systému, kde je spuštěn server mqweb, který je hostitelem serveru messaging REST API , vytvořte nebo aktualizujte soubor JSON CCDT, který obsahuje informace o připojení správce front.

Soubor CCDT musí obsahovat informace o `name`, `clientConnectiona` `type` . Volitelně můžete zahrnout další informace, jako např. informace `transmissionSecurity` . Další informace o všech definicích atributů kanálu CCDT naleznete v tématu Úplný seznam definic atributů kanálu CCDT.

Následující příklad ukazuje základní soubor JSON CCDT pro připojení vzdáleného správce front. Nastaví název kanálu na stejný název jako vzorový kanál připojení serveru vytvořený v kroku "1" na stránce 689. Port připojení je nastaven na stejnou hodnotu jako port používaný modulem listener. Hostitel připojení je nastaven na název hostitele systému, na kterém je spuštěn vzdálený správce front QM1.

```
{
  "channel": [{
      "name": "QM1.SVRCONN",
      "clientConnection": {
        "connection": [{
            "host": "example.com",
            "port": 1414
          }],
        "queueManager": "QM1"
      },
      "type": "clientConnection"
    }]
}
```

6. Z instalace, na které je spuštěn server mqweb, který je hostitelem serveru messaging REST API, použijte příkaz **setmqweb remote** k přidání informací o vzdáleném správci front do konfigurace serveru mqweb.

Je třeba zadat alespoň následující parametry:

- **-qmgrName**, kde zadáte název správce front.
- **-ccdtURL**, kde určíte URL tabulky CCDT pro správce front.
- **-uniqueName**, kde zadáte jedinečný název pro správce front. Jedinečný název se používá k odlišení vzdálených správců front, kteří mohou mít stejný název, a proto nesmí existovat k identifikaci jiného správce front.

Můžete zadat několik dalších voleb, například jméno uživatele a heslo, které se mají použít pro připojení vzdáleného správce front, nebo podrobnosti o úložišti údajů o důvěryhodnosti a úložišti klíčů. Úplný seznam parametrů, které lze zadat pomocí příkazu **setmqweb remote** , naleznete v části setmqweb remote.

Chcete-li například přidat vzdáleného správce front QM1s ukázkovým souborem CCDT, zadejte následující příkaz:

```
setmqweb remote add -uniqueName "RemoteQM1" -qmgrName "QM1" -ccdtURL "c:\myccdts\ccdt.json"
```

Výsledky

Vzdáleného správce front lze použít s produktem messaging REST API pomocí jedinečného názvu v prostředku URL namísto názvu správce front.

Příklad

Následující příklad nastaví připojení vzdáleného správce front pro správce front QM1. Konzola IBM MQ Console autorizovaná pro administraci správce front na základě autorizace poskytnuté uživateli exampleUser. Pověření tohoto uživatele jsou poskytnuta produktu IBM MQ Console , když je produkt **setmqweb remote** použit ke konfiguraci připojení správce front.

1. V systému, kde je vzdálený správce front QM1 , se vytvoří kanál připojení serveru a modul listener. Modul listener je spuštěn a uživateli `exampleUser` je poskytnuta autorizace pro připojení ke správci front a přístup k frontě, která se používá pro systém zpráv:

```
runmqsc QM1
#Define the server connection channel that will accept connections from the Console
DEFINE CHANNEL(QM1.SVRCONN) CHLTYPE(SVRCONN) TRPTYPE(TCP)
# Define the listener to use for the connection from the Console
DEFINE LISTENER(REMOTE.LISTENER) TRPTYPE(TCP) PORT(1414)
# Start the listener
START LISTENER(REMOTE.LISTENER)
end

#Set mq authorization for exampleUser to access the queue manager and a queue for messaging
setmqaut -m QM1 -t qmgr -p exampleUser +connect +inq +setall +dsp
setmqaut -m QM1 -t queue -p exampleUser -n EXAMPLEQ +put +get +browse +inq
```

2. V systému, kde je spuštěn server mqweb, je vytvořen soubor `QM1_ccdt.json` s následujícími informacemi o připojení:

```
{
  "channel": [{
      "name": "QM1.SVRCONN",
      "clientConnection": {
        "connection": [{
            "host": "example.com",
            "port": 1414
          }],
        "queueManager": "QM1"
      },
      "type": "clientConnection"
    }]
}
```

3. V systému, kde je spuštěn server mqweb, jsou na server mqweb přidány informace o připojení pro správce front QM1 . Pověření pro `exampleUser` jsou zahrnuta v informacích o připojení:

```
setmqweb remote add -uniqueName "MACHINEAQM1" -qmgrName "QM1" -ccdtURL
"c:\myccdts\QM1_ccdt.json" -username "exampleUser" -password "password"
```

4. Produkt messaging REST API se může připojit ke vzdálenému správci front QM1 pomocí jedinečného názvu pro připojení správce front namísto názvu správce front v prostředku URL:

```
curl -k https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/MACHINEAQM1/queue/EXAMPLEQ/
message -X POST -u myuser:mypassword -H "ibm-mq-rest-csrf-token: value" -H "Content-Type:
text/plain;charset=utf-8" --data "Hello World!"
```

V 9.3.3 Nastavení skupiny správců front pro použití s rozhraním REST API systému zpráv

Pomocí konzoly messaging REST API se můžete připojit ke skupinám správců front pro systém zpráv. Než se budete moci připojit ke skupině správců front, musíte pro tuto skupinu nastavit konfiguraci vzdáleného správce front. Poté se můžete připojit ke skupině správců front pomocí jedinečného názvu, který je definován v informacích o konfiguraci.

Než začnete

- Ujistěte se, že jste nakonfigurovali server mqweb pro použití administrative REST API, administrative REST API pro MFT, messaging REST APInebo IBM MQ Console.Další informace o konfiguraci serveru mqweb se základním registrem naleznete v tématu Základní konfigurace serveru mqweb.
- Pokud je server mqweb již nakonfigurován, ujistěte se, že jste přidali příslušné uživatele pro povolení systému zpráv v kroku 5 Základní konfigurace pro server mqweb. Uživatelé produktu messaging REST API musí být členem role `MQWebUser` . Role `MQWebAdmin` a `MQWebAdminRO` nelze použít pro messaging REST API.
  - Je-li parametr **`mqRestMessagingAdoptWebUserContext`** v konfiguraci serveru mqweb nastaven na hodnotu `true` , musí být uživatelé v roli `MQWebUser` autorizováni pro přístup k frontám a tématům, která se používají pro systém zpráv. Tyto uživatele můžete autorizovat pomocí OAM nebo RACF.
  - Je-li parametr **`mqRestMessagingAdoptWebUserContext`** v konfiguraci serveru mqweb nastaven na hodnotu `false` , musí být ID uživatele, který spouští server mqweb, autorizován pro přístup k frontám a tématům, která se používají pro systém zpráv. Tohoto uživatele můžete autorizovat pomocí OAM nebo RACF.
- Ujistěte se, že je agent messaging REST API konfigurován pro připojení ke vzdáleným správcům front. Další informace viz Konfigurace režimu připojení pro messaging REST API

Informace o této úloze

Skupina správců front umožňuje připojení aplikací k libovolnému správci front v rámci skupiny. Skupina je definována jako sada připojení v tabulce CCDT (Client Channel Definition Table). Při použití volání MQCONN nebo MQCONNX odkazujete na skupinu přidáním předpony hvězdičky k názvu správce front. V produktu messaging REST APIodkazujete na skupinu pomocí jedinečného názvu, který je přidružen ke skupině správců front. Jedinečný název je obsažen v adrese URL prostředku namísto názvu správce front. Další informace o skupinách správců front viz “Skupiny správců front v tabulce CCDT” na stránce 892.

Můžete také konfigurovat vzdálené správce front jednotlivě. Další informace viz téma “Nastavení vzdáleného správce front pro použití s produktem messaging REST API” na stránce 688.

Postup

1. V každém ze vzdálených správců front ve skupině vytvořte kanál připojení serveru, který umožní vzdálená připojení ke správci front. K vytvoření kanálů připojení serveru můžete použít příkaz **`DEFINE CHANNEL`** MQSC na příkazovém řádku.
   Chcete-li například vytvořit kanál připojení serveru `QM1.SVRCONN` pro správce front QM1, zadejte následující příkaz:

   ```
   DEFINE CHANNEL(QM1.SVRCONN) CHLTYPE(SVRCONN) TRPTYPE(TCP)
   ```

   Další informace o produktu **`DEFINE CHANNEL`** a dostupných volbách viz DEFINE CHANNEL.

2. Ve všech vzdálených správcích front ve skupině zkontrolujte, zda je příslušný uživatel autorizován pro přístup ke správci front. Tento uživatel musí být také autorizován pro přístup ke všem frontám nebo tématům, které používáte pro zasílání zpráv. Uživatel potřebuje oprávnění `connect`, `inquire`, `alternate usera set context` ke správci front. V systému UNIX, Linux, and Windows použijte řídicí příkaz **setmqaut** na příkazovém řádku. V systému z/OSdefinujte profily RACF , které poskytnou autorizovanému uživateli přístup ke správci front.
   Například v systému UNIX, Linux, and Windowszadejte následující příkaz pro autorizaci uživatele `exampleUser`pro přístup ke správci front QM1:

   ```
   setmqaut -m QM1 -t qmgr -p exampleUser +connect +inq +altusr +setall
   ```

   Další informace o tom, který uživatel musí být autorizován, viz "Určení činitele zabezpečení používaného produktem messaging REST API" na stránce 694.

3. ALW
   Pokud ve všech vzdálených správcích front ve skupině neexistuje žádný modul listener, vytvořte moduly listener pro příjem příchozích síťových připojení. K vytvoření modulů listener můžete použít příkaz **DEFINE LISTENER** MQSC na příkazovém řádku.
   Chcete-li například vytvořit modul listener `REMOTE.LISTENER` na portu 1414 pro vzdáleného správce front QM1, zadejte následující příkaz:

   ```
   runmqsc QM1
   DEFINE LISTENER(REMOTE.LISTENER) TRPTYPE(TCP) PORT(1414)
   end
   ```

4. V každém ze vzdálených správců front ve skupině zkontrolujte, zda je modul listener spuštěn pomocí příkazu **START LISTENER** MQSC na příkazovém řádku.

   ALW Chcete-li například v systému AIX, Linux, and Windows spustit modul listener `REMOTE.LISTENER` pro správce front QM1, zadejte následující příkaz:

   ```
   runmqsc QM1
   START LISTENER(REMOTE.LISTENER)
   end
   ```

   z/OS Chcete-li například v systému z/OSspustit modul listener, zadejte následující příkaz:

   ```
   runmqsc QM1
   START LISTENER TRPTYPE(TCP) PORT(1414)
   end
   ```

   Před spuštěním modulu listener v systému z/OSmusí být spuštěn adresní prostor inicializátoru kanálu.

5. V systému, kde je spuštěn server mqweb, který hostuje server messaging REST API , vytvořte soubor JSON CCDT. Tento soubor JSON obsahuje informace o připojení pro každého správce front ve skupině.

   Soubor CCDT musí obsahovat informace o `name`, `clientConnectiona type` pro každé připojení správce front. Volitelně můžete zahrnout další informace, jako např. informace `transmissionSecurity` . Další informace o všech definicích atributů kanálu CCDT naleznete v tématu Úplný seznam definic atributů kanálu CCDT.

   Následující příklad ukazuje základní soubor JSON CCDT pro dvě připojení správce front. První připojení je určeno pro správce front QM1. Má kanál připojení serveru `QM1.SVRCONN`, modul listener na portu 1414a běží na hostiteli `QM1.example.com`. Druhé připojení je určeno pro správce front QM2. Má kanál připojení serveru `QM2.SVRCONN`, modul listener na portu 1415a běží na hostiteli `QM2.example.com`. Jelikož jsou však připojení součástí skupiny správců front QMGRP, je pole **queueManager** pro obě připojení nastaveno na název skupiny správců front.

   ```
   {
     "channel": [{
         "name": "QM1.SVRCONN",
         "clientConnection": {
           "connection": [{
               "host": "QM1.example.com",
               "port": 1414
   ```

```
          }],
        "queueManager": "QMGRP"
      },
      "type": "clientConnection"
   }],
   "channel": [{
      "name": "QM2.SVRCONN",
      "clientConnection": {
        "connection": [{
            "host": "QM2.example.com",
            "port": 1415
          }],
        "queueManager": "QMGRP"
      },
      "type": "clientConnection"
    }]
}
```

6. V instalaci, na které je spuštěn server mqweb, který je hostitelem serveru messaging REST API, přidejte pomocí příkazu **setmqweb remote** skupinu správců front do konfigurace serveru mqweb.

   Je třeba zadat alespoň následující parametry:

   - **-qmgrName**, kde zadáte název skupiny pro skupinu správců front.
   - **-ccdtURL**, kde určíte URL tabulky CCDT pro správce front.
   - **-uniqueName**, kde zadáte jedinečný název pro identifikaci skupiny správců front.
   - **-group**, chcete-li nastavit informace o správci front jako skupinu.

   Můžete uvést několik dalších voleb, například jméno uživatele a heslo, které se mají použít pro připojení, nebo podrobnosti úložiště údajů o důvěryhodnosti a úložiště klíčů. Úplný seznam parametrů, které lze zadat pomocí příkazu **setmqweb remote** , naleznete v části setmqweb remote.

   Chcete-li například přidat skupinu správců front QMGRP s ukázkovým souborem CCDT, zadejte následující příkaz:

```
setmqweb remote add  -uniqueName "MyQMGRP" -qmgrName "QMGRP" -ccdtURL
"c:\myccdts\group_ccdt.json" -group
```

Výsledky

Skupinu vzdálených správců front lze použít s produktem messaging REST API s použitím jedinečného názvu v adrese URLprostředku. Správce front ze skupiny je vybrán k dokončení požadavku a informace o tom, který správce front dokončil požadavek, jsou vráceny v záhlaví odezvy `ibm-mq-resolved-qmgr`.

V 9.3.3 Určení činitele zabezpečení používaného produktem messaging REST API

Při použití konzoly messaging REST APImusí být příslušný uživatel autorizován pro přístup ke správcům front, frontám a tématům, ke kterým se chcete připojit pro systém zpráv. Uživatel, kterého je třeba autorizovat, závisí na konfiguraci serveru mqweb a na tom, zda používáte vzdálené správce front s produktem messaging REST API.

Ve výchozím nastavení je činitelem zabezpečení, který se používá k autorizaci přístupu ke správci front, uživatel, který spouští server mqweb, na kterém běží produkt messaging REST API. Činitel zabezpečení, který se používá k autorizaci přístupu k frontám a tématům, je uživatel, který je přihlášen k serveru messaging REST API. Připojení serveru mqweb nebo vzdáleného správce front však může být konfigurováno tak, že bude použit jiný činitel zabezpečení.

Určení činitele zabezpečení, který se používá pro připojení ke správci front

Pro připojení lokálního správce front je činitelem zabezpečení, který se používá pro připojení ke správci front, uživatel, který spouští server mqweb, na kterém běží produkt messaging REST API. V případě připojení vzdáleného správce front jsou následující činitelé zabezpečení používáni produktem messaging REST API k autorizaci přístupu ke správci front v pořadí podle priority. To znamená, že pokud jsou

uživatelé určeni více způsoby v rámci konfigurace vzdáleného správce front, první v seznamu se použije pro autorizaci.

1. Činitel zabezpečení je adoptovaný uživatelský kontext z uživatelské procedury zabezpečení.
2. Činitel zabezpečení je adoptovaný uživatelský kontext v pravidle CHLAUTH v kanálu připojení serveru, který se používá pro připojení ke vzdálenému správci front.
3. Činitel zabezpečení je ID uživatele, které je zahrnuto v konfiguraci vzdáleného správce front pro produkt messaging REST API. Toto ID uživatele je volitelně zahrnuto v informacích o připojení správce front při přidávání správce front pomocí příkazu **setmqweb remote** .
4. Činitel zabezpečení je uživatel, který spouští server mqweb, na kterém běží produkt messaging REST API.

Další informace o nastavení vzdálených správců front pro použití s produktem messaging REST APInaleznete v části "Nastavení vzdáleného správce front pro použití s produktem messaging REST API" na stránce 688.

Určení činitele zabezpečení, který se používá pro připojení k frontám a tématům

Můžete nastavit vlastnost v konfiguraci serveru mqweb a určit, který činitel zabezpečení se používá k autorizaci připojení k frontám a tématům při použití konzoly messaging REST API. Tato vlastnost je vlastností **mqRestMessagingAdoptWebUserContext** . Pomocí příkazu **dspmqweb properties** můžete zobrazit, na co je tato vlastnost nastavena.

- Je-li parametr **mqRestMessagingAdoptWebUserContext** nastaven na hodnotu true, použije produkt messaging REST API pro autorizaci ID uživatele, který je přihlášen k produktu messaging REST API . Proto ID uživatele nebo ID uživatele, která existují v konfiguraci serveru mqweb pro použití s produktem messaging REST API , jsou činitelé zabezpečení, kteří musí být autorizováni pro přístup k frontám a tématům.
- Je-li parametr **mqRestMessagingAdoptWebUserContext** nastaven na hodnotu false, použije agent messaging REST API pro autorizaci ID uživatele, který spustil server mqweb, který je hostitelem produktu messaging REST API . Proto musí být ID uživatele, které je stejné jako ID uživatele, který spouští server mqweb, který je hostitelem serveru messaging REST API , autorizováno pro přístup k frontám a tématům.

  Pokud se vaše fronty a témata nacházejí ve vzdáleném správci front, činitel zabezpečení, který se používá pro autorizaci, může být určen nastavením v konfiguraci správce front. V pořadí podle priority mohou být použity následující řídicí služby zabezpečení:

  1. Činitel zabezpečení je adoptovaný uživatelský kontext z uživatelské procedury zabezpečení.
  2. Činitel zabezpečení je adoptovaný uživatelský kontext v pravidle CHLAUTH v kanálu připojení serveru, který se používá pro připojení ke vzdálenému správci front. Můžete například nakonfigurovat pravidlo CHLAUTH v kanálu připojení serveru tak, aby používalo parametr MCAUSER. Poté jsou všechna připojení mapována na ID uživatele, které je autorizováno používat správce front.
  3. Činitel zabezpečení je adoptovaný uživatelský kontext z parametru AUTHINFO správce front. Pokud je objekt AUTHINFO, na který odkazuje atribut CONNAUTH správce front, konfigurován pro použití produktu **ADOPTCTX(yes)**, bude činitel zabezpečení, který se používá k autorizaci připojení ke správci front, použit také k autorizaci front a témat. Tento činitel zabezpečení může být například ID uživatele, které je zahrnuto v informacích o připojení vzdáleného správce front v rámci příkazu **setmqweb remote** .

Související informace

CHLAUTH

CONNAUTH

Vlastnosti příkazu dspmqweb

Vývoj aplikací MQI pomocí produktu 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 své aplikace ve zvoleném jazyce, naleznete v dílčích tématech.

Přehled rozhraní volání pro procedurální jazyky viz 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 definice dat, které vám pomohou při psaní aplikací. Úplný popis viz “Soubory definice dat IBM MQ” na stránce 696.

Chcete-li vám pomoci vybrat, ve kterém procedurálním jazyce mají být programy kódovány, zvažte maximální délku zpráv, které budou vaše programy zpracovávat. Pokud vaše programy zpracují pouze zprávy o známé maximální délce, můžete je kódovat v libovolném podporovaném jazyce. Pokud neznáte maximální délku zpráv, které budou programy muset zpracovat, bude zvolený jazyk záviset na tom, zda píšete CICS, IMSnebo dávkovou aplikaci:

IMS a dávka
: Programujte programy v jazyce C, PL/I nebo assembleru tak, aby používaly prostředky, které tyto jazyky nabízejí pro získání a uvolnění libovolného množství paměti. Můžete také kódovat své programy v jazyce COBOL, ale k získání a uvolnění paměti můžete použít podprogramy assembleru, PL/I nebo C.

CICS
: Kódujte programy v libovolném jazyce podporovaném produktem CICS. Rozhraní EXEC CICS poskytuje v případě potřeby volání pro správu paměti.

Související pojmy

“Objektově orientované aplikace” na stránce 15
Produkt IBM MQ poskytuje podporu pro JMS, Java, C + + a .NET. Tyto jazyky a rámce používají objektový model IBM MQ , který poskytuje třídy, které poskytují stejné funkce jako volání a struktury produktu IBM MQ .

Technický přehled

“Koncepty vývoje aplikací” na stránce 7
K psaní aplikací IBM MQ můžete použít výběr procedurálních nebo objektově orientovaných jazyků. Než začnete navrhovat a psát aplikace IBM MQ , seznamte se se základními koncepty produktu IBM MQ .

Související odkazy

Odkaz na vývoj aplikací

Soubory definice dat IBM MQ

Produkt IBM MQ poskytuje soubory definice dat, které vám pomohou při psaní aplikací.

Soubory definice dat jsou také známé jako:

Jazyk	Definice dat
C	Zahrnout soubory nebo soubory záhlaví
Visual Basic	Soubory modulu (pouze 32bitové verze)
COBOL	Kopírovat soubory
Sestavovací modul	Makra
PL/I	Zahrnout soubory

Soubory definice dat, které vám pomohou zapsat uživatelské procedury kanálu, jsou popsány v části IBM MQ COPY, soubory záhlaví, zahrnutí a soubory modulu.

Soubory definic dat, které vám pomohou při zápisu instalovatelných uživatelských procedur služeb, jsou popsány v tématu “Uživatelské procedury, uživatelské procedury rozhraní API a instalovatelné služby IBM MQ” na stránce 903.

Soubory definice dat podporované v jazyku C + + viz Použití C++.

IBM i

Soubory definic dat podporované v RPG naleznete v příručce IBM i Application Programming Reference (ILE/RPG).

Názvy souborů definic dat mají předponu CMQ a příponu určenou programovacím jazykem:

Přípona	Jazyk
a	Jazyk asembleru
b	Visual Basic
c	C
l	COBOL (bez inicializovaných hodnot)
p	PL/I
v	COBOL (s nastavenými výchozími hodnotami)

Instalační knihovna

Název **thlqual** je kvalifikátor vyšší úrovně instalační knihovny v systému z/OS.

Toto téma uvádí soubory definice dat IBM MQ pod těmito nadpisy:

- “Soubory začlenění jazyka C” na stránce 697
- “Soubory modulu Visual Basic” na stránce 698
- “Kopírovat soubory v jazyce COBOL” na stránce 698
- z/OS “System/390 makra v jazyce assembleru” na stránce 699
- z/OS “Soubory začlenění PL/I” na stránce 699

Soubory začlenění jazyka C

IBM MQ Zahrnout soubory C jsou uvedeny v Hlavičkových souborech C. Jsou nainstalovány v následujících adresářích nebo knihovnách:

Platforma	Instalační adresář nebo knihovna
IBM i IBM i	QMQM/H
Linux AIX AIX and Linux	*MQ_INSTALLATION_PATH*/včetně/
Windows Windows	*MQ_INSTALLATION_PATH*\Tools\c\include
z/OS z/OS	**thlqual**.SCSQC370

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

Poznámka: V případě systému AIX and Linuxjsou soubory začlenění symbolicky propojeny s `/usr/include`.

Další informace o struktuře adresářů naleznete v tématu Plánování podpory systému souborů.

Soubory modulu Visual Basic

Produkt IBM MQ for Windows poskytuje čtyři soubory modulu Visual Basic.

Jsou uvedeny v seznamu Soubory modulu Visual Basic a instalovány v

```
MQ_INSTALLATION_PATH\Tools\Samples\VB\Include
```

Kopírovat soubory v jazyce COBOL

Pro COBOL IBM MQ poskytuje oddělené kopírované soubory obsahující pojmenované konstanty a dva kopírované soubory pro každou ze struktur.

Pro každou strukturu existují dva kopírované soubory, protože každý z nich má i bez počátečních hodnot:

- V WORKING-STORAGE SECTION programu v jazyce COBOL použijte soubory, které inicializují pole struktury na výchozí hodnoty. Tyto struktury jsou definovány v kopírovaných souborech, které mají názvy s příponou písmene V (hodnoty).
- V sekci LINKAGE SECTION programu v jazyce COBOL použijte struktury bez počátečních hodnot. Tyto struktury jsou definovány v kopírovaných souborech, které mají názvy s příponou L (linkage).

IBM i Kopírované soubory obsahující definice dat a rozhraní pro produkt IBM i jsou k dispozici pro programy ILE COBOL, které používají prototypovaná volání 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).

IBM MQ Soubory kopií v jazyce COBOL jsou uvedeny v seznamu Soubory COPY v jazyce COBOL. Jsou nainstalovány v následujících adresářích:

Platforma	Instalační adresář nebo knihovna
Linux AIX AIX and Linux	*MQ_INSTALLATION_PATH*/včetně/
IBM i IBM i	QMQM/QCBLLESRC
Windows Windows	*MQ_INSTALLATION_PATH*\Tools\cobol\copybook (pro Micro Focus COBOL) *MQ_INSTALLATION_PATH*\Tools\cobol\copybook\VAcobol (pro IBM VisualAge COBOL)
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. Proveďte to s jedním nebo více příkazy COPY po deklaraci level-01 . To znamená, že v případě potřeby můžete do programu zahrnout více verzí struktur. Všimněte si, že CMQV je velký soubor.

Zde je příklad kódu COBOL pro zahrnutí kopírovacího souboru CMQMDV:

```
01 MQM-MESSAGE-DESCRIPTOR.
COPY CMQMDV.
```

Každá deklarace struktury začíná položkou level-01 . Můžete deklarovat několik instancí struktury tak, že označíte deklaraci level-01 následovanou příkazem COPY, který se má zkopírovat do zbytku deklarace struktury. Chcete-li se odkázat na příslušnou instanci, použijte klíčové slovo IN.

Zde je příklad kódu COBOL, který zahrnuje dvě instance 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.
```

Zarovnejte struktury na 4bajtových hranicích. Pokud použijete příkaz COPY k zahrnutí struktury za položku, která není položkou level-01 , ujistěte se, že je struktura násobkem 4 bajtů od začátku položky level-01 . Pokud tak neučiníte, můžete snížit výkon aplikace.

Struktury jsou popsány v části Datové typy používané v rozhraní MQI. Popisy polí ve strukturách zobrazují názvy polí bez předpony. V programech v COBOLu uveďte před názvy polí název struktury následovaný pomlčkou, jak je zobrazeno v deklaracích v COBOLu. Pole v souborech kopie struktury mají tímto způsobem předponu.

Názvy polí v deklaracích v souborech kopie struktury jsou uvedeny velkými písmeny. Místo toho můžete použít malá nebo velká písmena. Například pole *StrucId* struktury MQGMO je zobrazeno jako MQGMO-STRUCID v deklaraci COBOL a v kopírovaném souboru.

Struktury přípon V jsou deklarovány s počátečními hodnotami pro všechna pole, takže musíte nastavit pouze ta pole, kde se požadovaná hodnota liší od počáteční hodnoty.

System/390 makra v jazyce assembleru

z/OS

Produkt IBM MQ for z/OS poskytuje dvě makra v jazyku asembleru obsahující pojmenované konstanty a jedno makro pro generování každé struktury.

Jsou uvedeny v souboru z/OS Assembler COPY a instalovány v adresáři **thlqual**.SCSQMACS.

Tato makra jsou volána pomocí tohoto kódu:

```
MY_MQMD CMQMDA EXPIRY=0,MSGTYPE=MQMT_DATAGRAM
```

Soubory začlenění PL/I

z/OS

Produkt IBM MQ for z/OS poskytuje soubory začlenění, které obsahují všechny definice, které potřebujete při psaní aplikací IBM MQ v PL/I.

Soubory jsou uvedeny v adresáři PL/I include files a instalovány v adresáři **thlqual**.SCSQPLIC:

Zahrňte tyto soubory do svého programu, pokud se chystáte propojit stub IBM MQ s vaším programem (viz “Příprava programu ke spuštění” na stránce 989). Zahrnout pouze CMQP, pokud hodláte dynamicky propojit volání IBM MQ (viz “Dynamické volání stubu IBM MQ” na stránce 995). Dynamické propojení lze provádět pouze pro dávkové programy a programy IMS .

Psaní procedurální žádosti o zařazení do fronty

Prostřednictvím těchto informací můžete získat informace o psaní aplikací řazení do front, o připojování a odpojování od správce front, o publikování/odběru a o otevírání a zavírání objektů.

Chcete-li se dozvědět více o psaní aplikací, použijte následující odkazy:

- “Přehled rozhraní fronty zpráv” na stránce 700
- “Připojení ke správci front a odpojení od něj” na stránce 713
- “Otevírání a zavírání objektů” na stránce 720
- “Vkládání zpráv do fronty” na stránce 730
- “Získávání zpráv z fronty” na stránce 744
- “Zápis aplikací publikování/odběru” na stránce 782

- “Zjišťování a nastavení atributů objektu” na stránce 822
- “Potvrzení a zálohování jednotek práce” na stránce 825
- “Spuštění aplikací IBM MQ pomocí spouštěčů” na stránce 836
- “Práce s rozhraním MQI a klastry” na stránce 854
- z/OS “Používání a psaní aplikací na systému IBM MQ for z/OS” na stránce 858
- z/OS “IMS a IMS přemostění aplikací na IBM MQ for z/OS” na stránce 66

Související pojmy

“Koncepty vývoje aplikací” na stránce 7
K psaní aplikací IBM MQ můžete použít výběr procedurálních nebo objektově orientovaných jazyků. Než začnete navrhovat a psát aplikace IBM MQ , seznamte se se základními koncepty produktu IBM MQ .

“Vyvíjení aplikací pro IBM MQ” na stránce 5
Můžete vyvíjet aplikace pro odesílání a příjem zpráv a pro správu správců front a souvisejících prostředků. Produkt IBM MQ podporuje aplikace napsané v mnoha různých jazycích a rámcích.

“Aspekty návrhu pro aplikace IBM MQ” na stránce 47
Když jste se rozhodli, jak mohou vaše aplikace využívat platformy a prostředí, která máte k dispozici, musíte se rozhodnout, jak používat funkce nabízené produktem IBM MQ.

“Psaní procedurálních aplikací klienta” na stránce 882
Co potřebujete vědět, abyste mohli psát klientské aplikace na systému IBM MQ pomocí procedurálního jazyka.

“Sestavení procedurální aplikace” na stránce 967
Aplikaci IBM MQ můžete napsat v jednom z několika procedurálních jazyků a spustit ji na několika různých platformách.

“Zpracování chyb procedurálních programů” na stránce 1004
Tyto informace vysvětlují chyby související s voláními rozhraní MQI aplikací, a to buď při volání, nebo při doručení zprávy do konečného místa určení.

Související úlohy

“Použití ukázkových procedurálních programů IBM MQ” na stránce 1023
Tyto ukázkové programy jsou napsány v procedurálních jazycích a ukazují typické použití rozhraní MQI (Message Queue Interface). Programy IBM MQ na různých platformách.

Přehled rozhraní fronty zpráv

Informace o komponentách rozhraní MQI (Message Queue Interface).

Rozhraní fronty zpráv se skládá z následujících položek:

- *Volání* , jejichž prostřednictvím mohou programy přistupovat ke správci front a jeho prostředkům.
- *Struktury* , které programy používají k předávání dat správci front a k získávání dat ze správce front.
- *Základní datové typy* pro předávání dat správci front a získávání dat ze správce front.

z/OS IBM MQ for z/OS také dodává:

- Dvě další volání, jejichž prostřednictvím mohou dávkové programy z/OS potvrzovat a vracet změny.
- *Soubory definice dat* (někdy známé jako kopírované soubory, makra, soubory začlenění a soubory záhlaví), které definují hodnoty konstant dodávaných s produktem IBM MQ for z/OS.
- *Programy stubů* pro propojení úprav s vašimi aplikacemi.
- Sada ukázkových programů, které demonstrují, jak používat rozhraní MQI na platformě z/OS . Další informace o těchto ukázkách viz “Použití ukázkových programů pro z/OS” na stránce 1122.

IBM i IBM MQ for IBM i také dodává:

- *Soubory definice dat* (někdy známé jako kopírované soubory, makra, soubory začlenění a soubory záhlaví), které definují hodnoty konstant dodávaných s produktem IBM MQ for IBM i.
- Tři programy stub pro propojení úprav s aplikacemi ILE C, ILE COBOL a ILE RPG.
- Sada ukázkových programů, které demonstrují, jak používat rozhraní MQI na platformě IBM i .

Systémy AIX, Linux, and Windows také dodávají:

- Volání, jejichž prostřednictvím mohou systémové programy IBM MQ for AIX, Linux, and Windows potvrdit a vrátit zpět změny.
- *Zahrnout soubory* , které definují hodnoty konstant dodávaných na těchto platformách.
- *Soubory knihovny* pro propojení aplikací.
- Sada ukázkových programů, které demonstrují, jak používat rozhraní MQI na těchto platformách. Další informace o těchto ukázkách viz "Použití ukázkových programů na platformě Multiplatforms" na stránce 1024.
- Ukázkový zdrojový a spustitelný kód pro vazby na externí správce transakcí.

Další informace o rozhraní MQI naleznete v následujících odkazech:

- "Volání MQI" na stránce 702
- "Volání synchronizačních bodů" na stránce 702
- "Převod dat, datové typy, definice dat a struktury" na stránce 703
- "IBM MQ programy stub a soubory knihovny" na stránce 704
- "Parametry společné pro všechna volání" na stránce 709
- "Určení vyrovnávacích pamětí" na stránce 709
- z/OS "z/OS aspekty dávkového zpracování" na stránce 710
- "AIX and Linux zpracování signálů" na stránce 710

Související pojmy

"Připojení ke správci front a odpojení od něj" na stránce 713
Chcete-li používat programovací služby IBM MQ , musí mít program připojení ke správci front. Pomocí těchto informací se dozvíte, jak se připojit ke správci front a jak se od něj odpojit.

"Otevírání a zavírání objektů" na stránce 720
Tyto informace poskytují náhled na otevírání a zavírání objektů IBM MQ .

"Vkládání zpráv do fronty" na stránce 730
Pomocí těchto informací se dozvíte, jak vkládat zprávy do fronty.

"Získávání zpráv z fronty" na stránce 744
Pomocí těchto informací získáte informace o získávání zpráv z fronty.

"Zjišťování a nastavení atributů objektu" na stránce 822
Atributy jsou vlastnosti, které definují charakteristiky objektu IBM MQ .

"Potvrzení a zálohování jednotek práce" na stránce 825
Tyto informace popisují, jak potvrdit a vrátit zpět všechny obnovitelné operace get a put, které se vyskytly v transakci.

"Spuštění aplikací IBM MQ pomocí spouštěčů" na stránce 836
Získejte informace o spouštěčích a o tom, jak spouštět aplikace IBM MQ pomocí spouštěčů.

"Práce s rozhraním MQI a klastry" na stránce 854
Existují speciální volby pro volání a návratové kódy, které se týkají klastrování.

"Používání a psaní aplikací na systému IBM MQ for z/OS" na stránce 858
Aplikace IBM MQ for z/OS lze vytvořit z programů, které jsou spuštěny v mnoha různých prostředích. To znamená, že mohou využívat zařízení, která jsou k dispozici ve více než jednom prostředí.

"IMS a IMS přemostění aplikací na IBM MQ for z/OS" na stránce 66
Tyto informace vám pomohou psát aplikace IMS pomocí IBM MQ.

Volání MQI

Pomocí těchto informací získáte informace o voláních v rozhraní MQI (Message Queue Interface).

Volání v rozhraní MQI lze seskupit takto:

MQCONN, MQCONNX a MQDISC
Tato volání slouží k připojení programu ke správci front (s volbami nebo bez nich) a k odpojení programu od správce front. Pokud napíšete CICS programy pro z/OS, nemusíte tato volání používat. Doporučuje se je však použít, chcete-li aplikaci přenést na jiné platformy.

MQOPEN a MQCLOSE
Tato volání slouží k otevření a zavření objektu, například fronty.

MQPUT a MQPUT1
Tato volání slouží k vložení zprávy do fronty.

MQGET
Toto volání slouží k procházení zpráv ve frontě nebo k odebírání zpráv z fronty.

MQSUB, MQSUBRQ
Tato volání slouží k registraci odběru tématu a k vyžádání publikování odpovídajících odběru.

MQINQ
Pomocí tohoto volání se dotazujte na atributy objektu.

MQSET
Toto volání slouží k nastavení některých atributů fronty. Nelze nastavit atributy jiných typů objektů.

MQBEGIN, MQCMIT a MQBACK
Tato volání použijte, když je IBM MQ koordinátorem pracovní jednotky. MQBEGIN spustí pracovní jednotku. MQCMIT a MQBACK ukončí pracovní jednotku a buď potvrdí, nebo odvolá aktualizace provedené během pracovní jednotky. IBM i IBM i Regulátor vázaného zpracování se používá ke koordinaci globálních pracovních jednotek na systému IBM MQ for IBM i. Používají se příkazy pro nativní spuštění vázaného zpracování, potvrzení a odvolání.

MQCRTMH, MQBUFMH, MQMHBUF, MQDLTMH
Tato volání slouží k vytvoření manipulátoru zprávy, k převedení manipulátoru zprávy na vyrovnávací paměť nebo vyrovnávací paměť na manipulátor zprávy a k odstranění manipulátoru zprávy.

MQSETMP, MQINQMP, MQDLTMP
Tato volání slouží k nastavení vlastnosti zprávy v popisovači zprávy, k dotazu na vlastnost zprávy a k odstranění vlastnosti z popisovače zprávy.

MQCB, MQCB_FUNCTION, MQCTL
Tato volání slouží k registraci a řízení funkce zpětného volání.

MQSTAT
Pomocí tohoto volání načtěte informace o stavu o předchozích asynchronních operacích vložení.

Popis volání MQI viz Popisy volání .

Volání synchronizačních bodů

Pomocí těchto informací zjistíte informace o voláních synchronizačních bodů na různých platformách.

Volání synchronizačních bodů 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 z/OS sdělte správci front, že všechny operace MQGET a MQPUT od posledního synchronizačního bodu mají být učiněny trvalými (potvrzenými) nebo mají být vráceny zpět. Chcete-li potvrdit a vrátit zpět změny v jiných prostředích, postupujte takto:

CICS
Použijte příkazy jako EXEC CICS SYNCPOINT a EXEC CICS SYNCPOINT ROLLBACK.

IMS
Použijte prostředky synchronizačního bodu IMS , například GU (get unique) pro volání IOPCB, CHKP (checkpoint) a ROLB (rollback).

RRS
Podle potřeby použijte MQCMIT a MQBACK nebo SRRCMIT a SRRBACK. (Viz “Správa transakcí a obnovitelné služby správce prostředků” na stránce 829.)

Poznámka: 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 prostředky vázaného zpracování IBM i (například EXEC CICS SYNCPOINT).

IBM MQ volání na platformách AIX, Linux, and Windows

ALW

IBM MQ for AIX, Linux, and Windows poskytují volání MQCMIT a MQBACK.

Pomocí volání synchronizačních bodů v programech sdělte správci front, že všechny operace MQGET a MQPUT od posledního synchronizačního bodu mají být trvalé (potvrzené) nebo mají být vráceny zpět. Chcete-li potvrdit a odvolat změny v prostředí CICS , použijte příkazy jako EXEC CICS SYNCPOINT a EXEC CICS SYNCPOINT ROLLBACK.

Převod dat, datové typy, definice dat a struktury

Pomocí těchto informací získáte informace o převodech dat, základních datových typech, definicích dat IBM MQ a strukturách při použití rozhraní fronty zpráv.

Převod dat

Volání MQXCNVC (convert characters) převádí znaková data zprávy z jedné znakové sady na jinou. S výjimkou systému IBM MQ for z/OSse toto volání používá pouze z uživatelské procedury pro převod dat.

Syntaxi použitou při volání MQXCNVC naleznete v části MQXCNVC-Convert characters a “Zápis uživatelských procedur převodu dat” na stránce 950 , kde jsou uvedeny pokyny pro zápis a vyvolání uživatelských procedur převodu dat.

Základní datové typy

Pro podporované programovací jazyky poskytuje rozhraní MQI základní datové typy nebo nestrukturovaná pole.

Tyto datové typy jsou plně popsány v části Základní datové typy.

IBM MQ Definice dat

z/OS Produkt IBM MQ for z/OS dodává definice dat ve formě kopírovacích souborů v jazyce COBOL, maker v jazyku sestavení, jednoho souboru začlenění PL/I, jednoho souboru začlenění jazyka C a souborů začlenění jazyka C + +.

IBM i Produkt IBM MQ for IBM i poskytuje definice dat ve formě souborů kopie v jazyce COBOL, souborů kopie v jazyce RPG, souborů zahrnutí jazyka C a souborů začlenění jazyka C + +.

Soubory definice dat dodané s produktem IBM MQ obsahují:

- Definice všech konstant a návratových kódů IBM MQ
- Definice struktur a datových typů IBM MQ

- Definice konstant pro inicializaci struktur
- Funkční prototypy pro každé volání (pouze pro PL/I a jazyk C)

Úplný popis souborů definice dat IBM MQ viz “Soubory definice dat IBM MQ” na stránce 696.

Struktury

Struktury používané s voláními MQI uvedenými v části “Volání MQI” na stránce 702jsou dodávány v souborech definice dat pro každý z podporovaných programovacích jazyků.

IBM i z/OS IBM MQ for z/OS a IBM MQ for IBM i dodejte soubory, které obsahují konstanty, které můžete použít při vyplňování některých polí těchto struktur. Další informace o těchto definicích viz IBM MQ definice dat.

Souhrn struktur viz Datové typy struktury .

IBM MQ programy stub a soubory knihovny

Zde jsou uvedeny programy stub a soubory knihovny pro každou platformu.

Další informace o tom, jak používat programy stub a soubory knihovny při sestavování spustitelné aplikace, viz “Sestavení procedurální aplikace” na stránce 967. Informace o odkazování na soubory knihovny C++ viz Použití C++ *IBM MQ Použití C++*.

AIX *Soubory knihovny IBM MQ for AIX*

V systému IBM MQ for AIXmusíte propojit svůj program se soubory knihovny MQI dodanými pro prostředí, ve kterém spouštíte aplikaci, a s těmi, které poskytuje operační systém.

V nevláknové aplikaci vytvořte odkaz na jednu z následujících knihoven:

Tabulka 106. Soubory knihovny pro aplikace AIX bez podprocesů

Soubor knihovny	**Prostředí**
libmqm.a	Server pro C
libmqic.a a libmqm.a	Klient pro C
libmqmzf.a	Instalovatelné uživatelské procedury 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 IBM MQ pro podporu Micro Focus COBOL
libmqmcb.a	Server pro jazyk COBOL
libmqicb.a	Klient pro COBOL
libimqc23ia.a	Klient pro C++ (XLC 16)
libimqs23ia.a	Server pro C++ (XLC 16)
V 9.3.5 libimqc23ca.a	Klient pro C++ (XLC 17)
V 9.3.5 libimqs23ca.a	Server pro C++ (XLC 17)

V 9.3.5 Knihovny obsahující "ia" byly sestaveny pomocí kompilátoru XLC 16, zatímco knihovny s "ca" v názvu byly sestaveny pomocí kompilátoru XLC 17.

V aplikaci s podporou podprocesů vytvořte odkaz na jednu z následujících knihoven:

Tabulka 107. Soubory knihovny pro aplikace AIX s podporou podprocesů. Tabulka se dvěma sloupci vypisující soubory knihovny a prostředí pro každý soubor knihovny.	
Soubor knihovny	**Prostředí**
libmqm_r.a	Server pro C
libmqic_r.a a libmqm_r.a	Klient pro C
libmqmzf_r.a	Instalovatelné uživatelské procedury 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++ (XLC 16)
libimqs23ia_r.a	Server pro C++ (XLC 16)
V 9.3.5 libimqc23ca_r.a	Klient pro C++ (XLC 17)
V 9.3.5 libimqs23ca_r.a	Server pro C++ (XLC 17)

V 9.3.5 Knihovny obsahující "ia" byly sestaveny pomocí kompilátoru XLC 16, zatímco knihovny s "ca" v názvu byly sestaveny pomocí kompilátoru XLC 17.

Poznámka: Nelze vytvořit odkaz na více než jednu knihovnu. To znamená, že nemůžete současně odkazovat na knihovnu s podporou podprocesů i na knihovnu bez podpory podprocesů.

IBM i *Soubory knihovny IBM MQ for IBM i*

V produktu IBM MQ for IBM ipropojte svůj program se soubory knihovny MQI dodanými pro prostředí, ve kterém spouštíte aplikaci, kromě souborů poskytovaných operačním systémem.

Pro nezávitové aplikace:

Tabulka 108. Soubory knihovny pro aplikace IBM i bez podprocesů	
Soubor knihovny	**Prostředí**
LIBMQM	Servisní program serveru a klienta
LIBMQIC	Servisní program klienta
IMQB23I4	Základní servisní program C++
IMQS23I4	Servisní program serveru C++
LIBMQMZF	Instalovatelné uživatelské procedury pro C

V aplikaci se podprocesy:

Tabulka 109. Soubory knihovny pro aplikace IBM i s podporou podprocesů	
Soubor knihovny	**Prostředí**
LIBMQM_R	Servisní program serveru & klienta
IMQB23I4_R	Základní servisní program C++

Tabulka 109. Soubory knihovny pro aplikace IBM i s podporou podprocesů (pokračování)	
Soubor knihovny	**Prostředí**
IMQS23I4_R	Servisní program serveru C++
LIBMQMZF_R	Instalovatelné uživatelské procedury pro C
LIBMQIC_R	Servisní program klienta

V systému IBM MQ for IBM imůžete psát své aplikace v jazyce C + +. Chcete-li zjistit, jak propojit své aplikace v jazyce C + +, a chcete-li získat úplné podrobnosti o všech aspektech používání jazyka C + +, prohlédněte si téma Použití C++.

Linux *IBM MQ pro soubory knihovny Linux*

V systému IBM MQ for Linuxmusíte kromě souborů poskytnutých operačním systémem propojit svůj program se soubory knihovny MQI dodanými pro prostředí, ve kterém spouštíte aplikaci.

V nevláknové aplikaci vytvořte odkaz na jednu z následujících knihoven:

Tabulka 110. Soubory knihovny pro aplikace Linux bez podprocesů	
Soubor knihovny	**Prostředí**
libmqm.so	Server pro C
libmqic.so a libmqm.so	Klient pro C
libmqmzf.so	Instalovatelné uživatelské procedury 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 s podporou podprocesů vytvořte odkaz na jednu z následujících knihoven:

Tabulka 111. Soubory knihovny pro aplikace Linux s podporou podprocesů	
Soubor knihovny	**Prostředí**
libmqm_r.so	Server pro C
libmqic_r.so a libmqm_r.so	Klient pro C
libmqmzf_r.so	Instalovatelné uživatelské procedury 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: Nelze vytvořit odkaz na více než jednu knihovnu. To znamená, že nemůžete současně odkazovat na knihovnu s podporou podprocesů i na knihovnu bez podpory podprocesů.

Windows *Soubory knihovny IBM MQ for Windows*

V systému IBM MQ for Windowsmusíte propojit svůj program se soubory knihovny MQI dodanými pro prostředí, ve kterém spouštíte aplikaci, kromě těch, které poskytuje operační systém:

Tabulka 112. Soubory knihovny pro aplikace Windows

Soubor knihovny	**Prostředí**
MQ_INSTALLATION_PATH\Tools\Lib\mqm.lib	Server pro 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 (32bitové)
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.lib32	Podpora serveru TXSeries CICS pro C (32bitové)
MQ_INSTALLATION_PATH\Tools\Lib\mqccics4.lib32	Podpora klienta TXSeries CICS pro C (32bitové)
MQ_INSTALLATION_PATH\Tools\Lib\mqmzf.lib	Uživatelské procedury instalovatelných služeb pro C (32bitové)
MQ_INSTALLATION_PATH\Tools\Lib\mqmcbb.lib	Server pro IBM COBOL (32bitový)
MQ_INSTALLATION_PATH\Tools\Lib\mqmcb.lib	Server pro mikrofokus COBOL (32bitový)
MQ_INSTALLATION_PATH\Tools\Lib\mqiccbb.lib	Klient pro IBM COBOL (32bitový)
MQ_INSTALLATION_PATH\Tools\Lib\mqiccb.lib	Klient pro mikrofokus 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 pro 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 (64bitové)
MQ_INSTALLATION_PATH\Tools\Lib64\mqcxa.lib	Rozhraní XA klienta pro C (64bitové)
MQ_INSTALLATION_PATH\Tools\Lib64\mqicxa.lib	Klient MTS pro C (64bitový)
MQ_INSTALLATION_PATH\Tools\Lib64\mqmcbb.lib	Server pro IBM COBOL (64bitový)
MQ_INSTALLATION_PATH\Tools\Lib64\mqmcb.lib	Server pro mikrofokus COBOL (64bitový)
MQ_INSTALLATION_PATH\Tools\Lib64\mqiccbb.lib	Klient pro IBM COBOL (64bitový)
MQ_INSTALLATION_PATH\Tools\Lib64\mqiccb.lib	Klient pro mikrofokus COBOL (64bitový)
MQ_INSTALLATION_PATH\Tools\Lib64\imqs23vn.lib	Server pro C++ (64bitový)
MQ_INSTALLATION_PATH\Tools\Lib64\imqc23vn.lib	Klient pro C++ (64bitový)

Tabulka 112. Soubory knihovny pro aplikace Windows (pokračování)	
Soubor knihovny	**Prostředí**
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 programů .NET . Další informace viz “Kompilace programů IBM MQ .NET” na stránce 595 v sekci “Vývoj aplikací .NET” na stránce 539 .

Tyto soubory jsou dodávány pro kompatibilitu s předchozími vydáními:

```
mqic32.lib
mqic32xa.lib
```

z/OS *IBM MQ for z/OS programy stub*

Než budete moci spustit program napsaný s produktem IBM MQ for z/OS, musíte jej propojit s programem stub dodávaným s produktem IBM MQ for z/OS pro prostředí, ve kterém spouštíte aplikaci.

Program stub poskytuje první fázi zpracování vašich volání do požadavků, které může produkt IBM MQ for z/OS zpracovat.

Produkt IBM MQ for z/OS dodává následující programy stub:

CSQBSTUB
Program stub pro dávkové programy z/OS

CSQBRRSI
Program stub pro dávkové programy z/OS používající RRS prostřednictvím MQI

CSQBRSTB
Program stub pro dávkové programy z/OS používající přímo RRS

CSQCSTUB
Program stub pro programy CICS

CSQQSTUB
Program stub pro programy IMS

CSQXSTUB
Program stub pro distribuované fronty, které nejsouCICS ,

CSQASTUB
Program stub pro uživatelské procedury převodu dat

Upozornění: Pokud použijete jiný program stub, než který je uveden pro specifické prostředí, může mít nepředvídatelné výsledky.

Poznámka: Pokud používáte program stub CSQBRSTB, linkování s ATRSCSS z SYS1.CSSLIB. (SYS1.CSSLIB je také známá jako *Knihovna volitelných služeb*). Další informace o RRS viz “Správa transakcí a obnovitelné služby správce prostředků” na stránce 829.

Alternativně můžete dynamicky volat stub z vašeho programu. Tato technika je popsána v části “Dynamické volání stubu IBM MQ” na stránce 995.

V produktu IMSmůže být také nutné použít speciální modul jazykového rozhraní dodávaný produktem IBM MQ.

Nespouštějte aplikace, které jsou upraveny pomocí odkazů s CSQBSTUB a CSQQSTUB ve stejné oblasti IMS MPP. To může způsobit problémy, jako např. zprávy DFS3607I nebo CSQQ005E . První volání MQCONN v adresním prostoru určuje, které rozhraní je použito, a proto musí být transakce CSQQSTUB a CSQBSTUB spuštěny v různých oblastech zpráv IMS .

Parametry společné pro všechna volání
Existují dva typy parametrů, které jsou společné pro všechna volání: manipulátory a návratové kódy.

Použití popisovačů

Všechna volání MQI používají jeden nebo více *manipulátorů*. Ty identifikují správce front, frontu nebo jiný objekt, zprávu nebo odběr podle potřeby volání.

Aby mohl program komunikovat se správcem front, musí mít jedinečný identifikátor, podle kterého zná tohoto správce front. Tento identifikátor se nazývá *manipulátor připojení*, někdy označovaný jako *Hconn*. V případě programů CICS je manipulátor připojení vždy nulový. Pro všechny ostatní platformy nebo styly programů je manipulátor připojení vrácen voláním MQCONN nebo MQCONNX při připojení programu ke správci front. Programy předávají manipulátor připojení jako vstupní parametr, když používají ostatní volání.

Aby mohl program pracovat s objektem IBM MQ , musí mít jedinečný identifikátor, podle kterého tento objekt zná. Tento identifikátor se nazývá *popisovač objektu*, někdy označovaný jako *Hobj*. Popisovač je vrácen voláním MQOPEN, když program otevře objekt, aby s ním mohl pracovat. Programy předávají popisovač objektu jako vstupní parametr při použití následných volání MQPUT, MQGET, MQINQ, MQSET nebo MQCLOSE.

Podobně volání MQSUB vrací *manipulátor odběru* nebo *Hsub*, který se používá k identifikaci odběru v následných voláních MQGET, MQCB nebo MQSUBRQ a určitá volání, která zpracovávají vlastnosti zprávy, používají *manipulátor zprávy* nebo *Hmsg*.

Základní informace o návratových kódech

Kód dokončení a kód příčiny jsou vráceny jako výstupní parametry pro každé volání. Tyto kódy jsou souhrnně označovány jako *návratové kódy*.

Chcete-li zobrazit, zda je volání úspěšné, vrátí každé volání po dokončení volání *kód dokončení* . Kód dokončení je obvykle MQCC_OK indikující úspěch nebo MQCC_FAILED indikující selhání. Některá volání mohou vrátit přechodný stav MQCC_WARNING, který označuje dílčí úspěch.

Každé volání také vrátí *kód příčiny* , který zobrazuje příčinu selhání nebo částečného úspěchu volání. Existuje mnoho kódů příčiny pokrývajících takové okolnosti, jako je zaplnění fronty, nepovolení operací získání pro frontu a nedefinování konkrétní fronty pro správce front. Programy mohou použít kód příčiny k rozhodnutí, jak pokračovat. Mohou například vyzvat uživatele ke změně vstupních dat, poté provést volání znovu, nebo mohou uživateli vrátit chybovou zprávu.

Je-li kód dokončení MQCC_OK, je kód příčiny vždy MQRC_NONE.

Kódy dokončení a příčiny pro každé volání jsou uvedeny s popisem tohoto volání. Viz Popisy volání a vyberte odpovídající volání ze seznamu.

Podrobnější informace, včetně nápadů na nápravná opatření, viz:

- z/OS položky IBM MQ for z/OS zprávy, dokončení, a kódy příčiny pro IBM MQ for z/OS
- Zprávy a kódy příčiny pro všechny ostatní IBM MQ platformy

Určení vyrovnávacích pamětí
Správce front odkazuje na vyrovnávací paměti pouze v případě, že jsou vyžadovány. Pokud nevyžadujete vyrovnávací paměť pro volání nebo pokud má vyrovnávací paměť nulovou délku, můžete použít ukazatel na vyrovnávací paměť s hodnotou Null.

Při zadávání požadované velikosti vyrovnávací paměti vždy používejte délku dat.

Pokud použijete vyrovnávací paměť k uchování výstupu volání (například k uchování dat zprávy pro volání MQGET nebo hodnot atributů dotazovaných voláním MQINQ), správce front se pokusí vrátit kód příčiny, pokud určená vyrovnávací paměť není platná nebo je v úložišti jen pro čtení. Nemusí však být vždy schopen vrátit kód příčiny.

z/OS *z/OS aspekty dávkového zpracování*

Dávkové programy z/OS , které volají rozhraní MQI, mohou být buď ve stavu supervizor, nebo problém.

Musí však splňovat tyto podmínky:

- Musí být v režimu úlohy, ne v režimu bloku požadavku na službu (SRB).
- Musí být v režimu ASC (Primary address space control) (ne v režimu ASC Access Register).
- Nesmí být v režimu křížové paměti. Číslo primárního adresního prostoru (ASN) se musí rovnat sekundárnímu ASN a domovskému ASN.
- Nesmí být použity jako uživatelské programy MPF.
- Nelze zadržovat žádné zámky z/OS .
- V zásobníku FRR nemohou být žádné rutiny pro obnovu funkcí (FRR).
- Pro volání MQCONN nebo MQCONNX může být platný libovolný klíč slova stavu programu (za předpokladu, že je klíč kompatibilní s použitím úložiště, které je v klíči TCB), ale následná volání používající manipulátor připojení vrácený MQCONN nebo MQCONNX:
  - Musí mít stejný klíč PSW, který byl použit ve volání MQCONN nebo MQCONNX.
  - Musí mít přístupné parametry (pro zápis, je-li to vhodné) pod stejným klíčem PSW
  - Musí být vydáno pod stejnou úlohou (TCB), ale ne v žádné podúloze úlohy
- Mohou být ve 24bitovém nebo 31bitovém režimu adresování. Je-li však v platnosti 24bitový režim adresování, musí být adresy parametrů interpretovány jako platné 31bitové adresy.

Není-li některá z těchto podmínek splněna, může dojít ke kontrole programu. V některých případech volání selže a vrátí se kód příčiny.

Linux AIX *Aspekty produktu AIX and Linux*

Aspekty, které musíte mít na paměti při vývoji aplikací AIX and Linux .

Linux AIX *Systémové volání fork v systémech AIX and Linux*

Povšimněte si těchto aspektů při použití systémového volání větvení v aplikacích IBM MQ .

Pokud vaše aplikace chce použít `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 vaše aplikace chce vytvořit podřízený proces po provedení volání IBM MQ , musí kód aplikace použít `fork()` s `exec()` , aby se zajistilo, že podřízená instance bude novou instancí, a nikoli přesnou kopií nadřízené instance.

Pokud vaše aplikace nepoužívá `exec()`, volání rozhraní API IBM MQ provedené v podřízeném procesu vrátí hodnotu MQRC_ENVIRONMENT_ERROR.

Linux AIX *AIX and Linux zpracování signálů*

Obecně platí, že systémy AIX and Linux se přesunuly z prostředí bez podprocesů (proces) do prostředí s podporou podprocesů. V mnoha případech se signály a manipulace se signály, i když jsou podporovány, nehodí dobře do prostředí s podporou podprocesů a existují různá omezení.

Obecně platí, že systémy AIX and Linux se přesunuly z prostředí bez podprocesů (proces) do prostředí s podporou podprocesů. V prostředí bez vláken mohou být některé funkce implementovány pouze pomocí signálů, ačkoli většina aplikací nepotřebuje znát signály a obsluhu signálů. V prostředí s podporou podprocesů podporují primitiva založená na podprocesech některé funkce, které byly implementovány v prostředích bez podpory podprocesů pomocí signálů.

V mnoha případech se signály a manipulace se signály, i když jsou podporovány, nehodí dobře do prostředí s podporou podprocesů 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 pokouší zpracovat signály. Tradiční přístup ukládání a obnovy obslužných rutin

signálů (definovaných pro každý proces), který fungoval, když v rámci procesu existoval pouze jeden podproces provedení, nefunguje v prostředí s podporou podprocesů. Důvodem je, že mnoho podprocesů provedení se může pokoušet uložit a obnovit prostředek pro celý proces s nepředvídatelnými výsledky.

Linux AIX *Nevláknové aplikace*

Každá funkce MQI nastavuje pro signály vlastní obslužnou rutinu signálu. Obslužné rutiny uživatelů pro tyto obslužné rutiny jsou nahrazeny po dobu trvání volání funkce MQI. Ostatní signály mohou být zachyceny v normálním způsobem uživatelem napsané manipulátory.

Každá funkce MQI nastavuje vlastní obslužnou rutinu signálu pro signály:

SIGALRM
SIGBUS
ZNAMENÍ
SIGSEGV
SIGILL

Obslužné rutiny uživatelů pro tyto obslužné rutiny jsou nahrazeny po dobu trvání volání funkce MQI. Ostatní signály mohou být zachyceny v normálním způsobem uživatelem napsané manipulátory. Pokud obslužnou rutinu nenainstalujete, zůstanou na místě výchozí akce (například ignorovat, výpis jádra nebo ukončit).

Poté, co produkt IBM MQ zpracuje synchronní signál (SIGSEGV, SIGBUS, SIGFPE, SIGILL), pokusí se před provedením volání funkce MQI předat signál libovolné registrované obslužné rutině signálu.

Linux AIX *Aplikace s podporou podprocesů*

Podproces je považován za připojený k produktu IBM MQ z MQCONN (nebo MQCONNX) do MQDISC.

Synchronní signály

Synchronní signály vznikají ve specifickém vláknu.

Systémy AIX and Linux bezpečně umožňují nastavení obslužné rutiny signálu pro takové signály pro celý proces. Produkt IBM MQ však nastaví vlastní obslužnou rutinu pro následující signály v procesu aplikace, zatímco je jakýkoli podproces připojen k produktu IBM MQ:

SIGBUS
ZNAMENÍ
SIGSEGV
SIGILL

Pokud píšete vícevláknové aplikace, existuje pouze jeden ovladač signálu pro celý proces pro každý signál. Když produkt IBM MQ nastaví své vlastní synchronní obslužné rutiny signálu, uloží všechny dříve registrované obslužné rutiny pro každý signál. Poté, co produkt IBM MQ zpracuje jeden z uvedených signálů, produkt IBM MQ se pokusí volat obslužnou rutinu signálu, která byla v platnosti v době prvního připojení produktu 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, nesmí podprocesy aplikací pro tyto signály zřizovat obslužné rutiny signálů, pokud existuje možnost, že je k produktu IBM MQpřipojen také jiný podproces téhož procesu.

Poznámka: Když aplikace nebo knihovna middlewaru (spuštěná jako součást aplikace) vytvoří obslužnou rutinu signálu, zatímco je podproces připojen k produktu IBM MQ, musí obslužná rutina signálu aplikace volat odpovídající obslužnou rutinu IBM MQ během zpracování tohoto signálu.

Při zřizování a obnovování obslužných rutin signálu je obecným principem, že poslední obslužná rutina signálu, která má být uložena, musí být první, která má být obnovena:

- Když aplikace po připojení k produktu IBM MQvytvoří obslužnou rutinu signálu, musí být před odpojením aplikace od produktu IBM MQobnovena předchozí obslužná rutina signálu.

- Když aplikace zavede obslužnou rutinu signálu před připojením k produktu IBM MQ, musí se před obnovením obslužné rutiny signálu odpojit od produktu IBM MQ .

Poznámka: Nedodržení obecného principu, že poslední ovladač signálu, který má být uložen, musí být první, který má být obnoven, může vést k neočekávanému zpracování signálu v aplikaci a potenciálně ke ztrátě signálů aplikací.

Asynchronní signály

Produkt IBM MQ nepoužívá v aplikacích s podprocesy žádné asynchronní signály, pokud se nejedná o klientské aplikace.

Další aspekty pro klientské aplikace s podporou podprocesů

Produkt IBM MQ zpracovává následující signály během vstupu/výstupu na server. Tyto signály jsou definovány komunikačním zásobníkem. Když je podproces připojen ke správci front, nesmí aplikace pro tyto signály vytvořit obslužnou rutinu signálu:

SIGPIPE (pro TCP/IP)

Linux AIX *Další aspekty při použití obsluhy signálu AIX and Linux v rozhraní MQI*

Při použití rozhraní MQI pro zpracování signálů v systému AIX and Linuxexistují další aspekty pro aplikace rychlé cesty, volání funkcí rozhraní MQI v rámci obslužných rutin signálů, signály během volání rozhraní MQI, uživatelské procedury a instalovatelné služby a obslužné rutiny procedur VMS.

Rychlé (důvěryhodné) aplikace

Aplikace rychlého nasazení se spouštějí ve stejném procesu jako produkt IBM MQ , takže jsou spuštěny v prostředí s podporou podprocesů.

V tomto prostředí produkt IBM MQ zpracovává synchronní signály SIGSEGV, SIGBUS, SIGFPE a SIGILL. Všechny ostatní signály nesmí být doručeny do aplikace Fastpath, když je připojen k IBM MQ. Místo toho musí být blokovány nebo zpracovány aplikací. Pokud aplikace Fastpath takovou událost zachytí, musí být správce front zastaven a restartován, nebo může být ponechán v nedefinovaném stavu. Úplný seznam omezení pro aplikace Fastpath v rámci MQCONNX viz “Připojení ke správci front pomocí volání MQCONNX” na stránce 715.

Volání funkce MQI v rámci obslužných rutin signálů

V době, kdy jste v obslužné rutině signálu, nevolejte funkci MQI.

Pokud se pokusíte volat funkci MQI z obslužné rutiny signálu, zatímco je aktivní jiná funkce MQI, vrátí se MQRC_CALL_IN_PROGRESS. Pokud se pokusíte volat funkci MQI z obslužné rutiny signálu, zatímco není aktivní žádná jiná funkce MQI, je pravděpodobné, že někdy během operace dojde k selhání kvůli omezením operačního systému, kde lze z obslužné rutiny nebo v rámci obslužné rutiny zadat pouze výběrová volání.

U metod destruktoru C++, které mohou být volány automaticky během uživatelské procedury programu, nemusíte být schopni zastavit volání funkcí MQI. Ignorujte jakékoli chyby týkající se MQRC_CALL_IN_PROGRESS. Pokud obslužná rutina signálů volá funkci exit (), produkt IBM MQ provede jako obvykle odvolání nepotvrzených zpráv v synchronizačním bodu a zavře všechny otevřené fronty.

Signály během volání MQI

Funkce MQI nevrací kód EINTR ani žádný ekvivalent k aplikačním programům.

Dojde-li během volání MQI k signálu a obslužná rutina zavolá *return*, bude volání pokračovat, jako by se signál nestal. Příkaz MQGET nemůže být přerušen signálem pro okamžité vrácení řízení do aplikace. Chcete-li přerušit operaci MQGET, nastavte frontu na GET_DISABLED; případně použijte smyčku kolem volání MQGET s konečným časovým vypršením (MQGMO_WAIT se sadou gmo.WaitInterval) a použijte

obslužnou rutinu signálu (v prostředí bez podprocesů) nebo ekvivalentní funkci v prostředí s podprocesy k nastavení příznaku, který přeruší smyčku.

AIX V prostředí AIX produkt IBM MQ vyžaduje restartování systémových volání přerušených signály. Při vytváření vlastní obslužné rutiny signálu s sigaction (2) nastavte příznak SA_RESTART v poli sa_flags nové struktury akce, jinak produkt IBM MQ nemusí být schopen dokončit jakékoli volání přerušené signálem.

Uživatelské procedury a instalovatelné služby

Uživatelské procedury a instalovatelné služby, které jsou spuštěny jako součást procesu IBM MQ v prostředí s podporou podprocesů, mají stejná omezení jako pro aplikace rychlé cesty. Zvažte jejich trvalé připojení k produktu IBM MQ , a tedy nepoužívání signálů nebo volání operačního systému bez zajištění neporušenosti vláken.

Připojení ke správci front a odpojení od něj

Chcete-li používat programovací služby IBM MQ , musí mít program připojení ke správci front. Pomocí těchto informací se dozvíte, jak se připojit ke správci front a jak se od něj odpojit.

Způsob, jakým je toto připojení vytvořeno, závisí na platformě a prostředí, ve kterém program pracuje:

Multi **IBM MQ for Multiplatforms**

Programy spuštěné v těchto prostředích mohou používat volání MQCONN MQI pro připojení ke správci front a volání MQDISC pro odpojení od správce front. Alternativně mohou programy používat volání MQCONNX.

z/OS **IBM MQ for z/OS dávkové**

Programy spuštěné v tomto prostředí mohou používat volání MQCONN MQI pro připojení ke správci front a volání MQDISC pro odpojení od správce front. Alternativně mohou programy používat volání MQCONNX.

Dávkové programy z/OS se mohou postupně nebo souběžně připojovat k více správcům front ve stejném 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 na systému z/OS, viz Administrace IBM MQ for z/OS. Zapisovací programy pro zařazování zpráv do front IMS však musí používat volání MQCONN MQI k určení správce front, ke kterému se chtějí připojit. Mohou použít volání MQDISC k odpojení od tohoto správce front.

Po volání IMS , které vytvoří synchronizační bod, a před zpracováním zprávy pro jiného uživatele adaptér IMS zajistí, že aplikace zavře manipulátory a odpojí se od správce front. Viz téma “Synchronizační body v aplikacích IMS” na stránce 828.

Programy IMS se mohou postupně nebo souběžně připojovat k více správcům front ve stejném TCB.

z/OS **CICS Transakční server pro z/OS**

Programy systému CICS nemusí provádět žádnou práci pro připojení ke správci front, protože je připojen samotný systém CICS . Toto připojení se obvykle vytváří automaticky při inicializaci, ale můžete také použít transakci CKQC, která je dodávána s produktem IBM MQ for z/OS. Další informace o CKQC viz Administrace IBM MQ for z/OS.

Úlohy CICS se mohou připojit pouze ke správci front, ke kterému je oblast CICS připojena.

Programy CICS mohou také používat volání připojení a odpojení rozhraní MQI (MQCONN a MQDISC). Možná to budete chtít udělat, abyste mohli tyto aplikace přenést do jiných prostředí nežCICS s minimem překódování. Avšak tato volání *vždy* byla úspěšně dokončena v prostředí 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 v daném okamžiku podporováno pouze jedno připojení. Aplikace CICS musí před ukončením zadat volání MQCONN pro získání manipulátoru připojení a volání MQDISC.

Další informace o připojení a odpojení od správce front naleznete v následujících odkazech:

- “Připojení ke správci front pomocí volání MQCONN” na stránce 714
- “Připojení ke správci front pomocí volání MQCONNX” na stránce 715
- “Odpojení programů od správce front pomocí MQDISC” na stránce 719

Související pojmy
“Přehled rozhraní fronty zpráv” na stránce 700
Informace o komponentách rozhraní MQI (Message Queue Interface).

“Otevírání a zavírání objektů” na stránce 720
Tyto informace poskytují náhled na otevírání a zavírání objektů IBM MQ .

“Vkládání zpráv do fronty” na stránce 730
Pomocí těchto informací se dozvíte, jak vkládat zprávy do fronty.

“Získávání zpráv z fronty” na stránce 744
Pomocí těchto informací získáte informace o získávání zpráv z fronty.

“Zjišťování a nastavení atributů objektu” na stránce 822
Atributy jsou vlastnosti, které definují charakteristiky objektu IBM MQ .

“Potvrzení a zálohování jednotek práce” na stránce 825
Tyto informace popisují, jak potvrdit a vrátit zpět všechny obnovitelné operace get a put, které se vyskytly v transakci.

“Spuštění aplikací IBM MQ pomocí spouštěčů” na stránce 836
Získejte informace o spouštěčích a o tom, jak spouštět aplikace IBM MQ pomocí spouštěčů.

“Práce s rozhraním MQI a klastry” na stránce 854
Existují speciální volby pro volání a návratové kódy, které se týkají klastrování.

“Používání a psaní aplikací na systému IBM MQ for z/OS” na stránce 858
Aplikace IBM MQ for z/OS lze vytvořit z programů, které jsou spuštěny v mnoha různých prostředích. To znamená, že mohou využívat zařízení, která jsou k dispozici ve více než jednom prostředí.

“IMS a IMS přemostění aplikací na IBM MQ for z/OS” na stránce 66
Tyto informace vám pomohou psát aplikace IMS pomocí IBM MQ.

Připojení ke správci front pomocí volání MQCONN
Pomocí těchto informací se dozvíte, jak se připojit ke správci front pomocí volání MQCONN.

Obecně se můžete připojit buď ke specifickému správci front, nebo k výchozímu správci front:

- V případě systému IBM MQ for z/OSje v dávkovém prostředí určen výchozí správce front v modulu CSQBDEFV.
- Pro systém IBM MQ for Multiplatformsje výchozí správce front určen v souboru mqs.ini .

Případně se v prostředích z/OS MVS batch, TSO a RRS můžete připojit k libovolnému správci front v rámci skupiny sdílení front. Požadavek MQCONN nebo MQCONNX vybere některého z aktivních členů skupiny.

Když se připojujete ke správci front, musí být lokální vzhledem k úloze. Musí patřit do stejného systému jako aplikace IBM MQ .

V prostředí IMS musí být správce front připojen k řídicí oblasti IMS a k závislé oblasti, kterou program používá. Výchozí správce front je určen v modulu CSQQDEFV při instalaci produktu IBM MQ for z/OS .

V prostředí TXSeries CICS a TXSeries pro Windows a AIXmusí být správce front definován jako prostředek XA pro CICS.

Chcete-li se připojit k výchozímu správci front, zavolejte MQCONN a zadejte název, který se skládá zcela z mezer nebo začíná znakem null (X'00 ').

Aplikace musí být autorizována, aby se mohla úspěšně připojit ke správci front. Další informace viz Zabezpečení.

Výstup z MQCONN je:

- Manipulátor připojení (**Hconn**)
- Kód dokončení
- Kód příčiny

Použijte manipulátor připojení pro následná volání MQI.

Pokud kód příčiny označuje, že aplikace je již připojena k danému správci front, vrácený manipulátor připojení je 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 rozsah manipulátoru objektu (viz "Otevírání objektů pomocí volání MQOPEN" na stránce 721).

Popisy parametrů jsou uvedeny v popisu volání MQCONN v MQCONN.

Volání MQCONN se nezdaří, pokud je správce front při spuštění volání ve stavu uvedení do klidového stavu nebo pokud se správce front vypíná.

Rozsah MQCONN nebo MQCONNX

Oborem volání MQCONN nebo MQCONNX je obvykle podproces, který jej vydal. To znamená, že manipulátor připojení vrácený z volání je platný pouze v rámci podprocesu, který volání vydal. Pomocí popisovače lze kdykoli provést pouze jedno volání. Pokud se používá z jiného podprocesu, je odmítnut jako neplatný. Máte-li v aplikaci více podprocesů a každý z nich chce použít volání IBM MQ , musí každý z nich vydat příkaz MQCONN nebo MQCONNX.

Není nutné, aby každé volání bylo provedeno do stejného správce front, když proces provádí více volání MQCONN. V daném okamžiku však lze z podprocesu vytvořit pouze jedno připojení IBM MQ . Případně zvažte možnost "Sdílená (nezávislá na vláknu) připojení s MQCONNX" na stránce 716 povolit použití více připojení IBM MQ z jednoho podprocesu a připojení IBM MQ z libovolného podprocesu.[7]

Pokud je 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 zahrnuje volby pro řízení způsobu, jakým volání funguje.

Jako vstup pro MQCONNX můžete zadat název správce front z/OS nebo název skupiny sdílení front na z/OS sdílených systémech front. Volby pro řízení způsobu vytvoření připojení ke správci front jsou dodávány ve struktuře nazvané MQCNO.

Výstup z MQCONNX je:

- Manipulátor připojení (Hconn)
- Kód dokončení
- Kód příčiny

Manipulátor připojení použijete při následných voláních MQI.

Volby připojení nastavené v poli *Options* struktury MQCNO umožňují řídit několik atributů připojení. Zvláštní poznámka jsou následující skupiny možností:

[7] Při použití aplikací s podporou podprocesů se systémy IBM MQ for AIX or Linux musíte zajistit, aby aplikace měly dostatečnou velikost zásobníku pro podprocesy. Zvažte použití velikosti zásobníku 256 kB nebo větší, když aplikace s podporou podprocesů uskutečňují volání MQI buď samy, nebo s jinými popisovači signálu (například CICS).

- Volby vazby umožňují vytvoření *důvěryhodných aplikací* . Důvěryhodné aplikace znamenají, že se aplikace IBM MQ a lokální agent správce front stanou stejným procesem. Vzhledem k tomu, že proces agenta již nemusí používat rozhraní pro přístup ke správci front, stanou se tyto aplikace rozšířením správce front. Toto chování je vyžadováno zadáním volby `MQCNO_FASTPATH_BINDING` . Další informace o omezeních, která se vztahují na důvěryhodné aplikace, naleznete v části "Omezení pro důvěryhodné aplikace" na stránce 716.
- Volby sdílení manipulátoru umožňují vytvoření sdílených připojení. Sdílená připojení mohou sdílet manipulátory mezi různými podprocesy v rámci stejného procesu. Další informace o sdílených připojeních viz "Sdílená (nezávislá na vláknu) připojení s MQCONNX" na stránce 716.

MQCNO také umožňuje aplikaci řídit způsob ověřování připojení ke správci front. Ověřovací pověření lze zadat ve struktuře MQCSP, na kterou odkazuje struktura MQCNO.

Úplný popis parametrů volání MQCONNX a atributů připojení, které lze řídit, naleznete v tématu MQCONNX-Connect queue manager (extended).

Omezení pro důvěryhodné aplikace
Omezení, která platí pro důvěryhodné aplikace. Některá omezení platí pro všechny platformy a jiná jsou specifická pro platformu.

T

- Je třeba explicitně odpojit důvěryhodné aplikace od správce front.
- Před ukončením správce front pomocí příkazu **endmqm** musíte zastavit důvěryhodné aplikace.
- S volbou MQCNO_FASTPATH_BINDING nesmíte používat asynchronní signály a přerušení časovače (například `sigkill`).
- 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.
- Linux AIX V systémech AIX and Linux musíte použít mqm jako efektivní userID a groupID pro všechna volání MQI. Tato ID můžete změnit před provedením volání jiného typu než MQI, které vyžaduje ověření (například otevření souboru), ale před provedením dalšího volání MQI jej musíte změnit zpět na mqm.
- IBM i V systému IBM i:
  1. Důvěryhodné aplikace musí být spuštěny pod profilem uživatele QMQM. Nestačí, aby byl profil uživatele členem skupiny QMQM nebo aby program převzal oprávnění QMQM. Je možné, že profil uživatele QMQM nebude možné použít k přihlášení k interaktivním úlohám nebo že bude uveden v popisu úlohy pro úlohy spouštějící důvěryhodné aplikace. V tomto případě je jedním z přístupů použití funkcí rozhraní API pro výměnu profilů systému IBM i , QSYGETPH, QWTSETP a QSYRLSPH pro dočasnou změnu aktuálního uživatele úlohy na QMQM při spuštění programů systému IBM MQ . Podrobnosti o těchto funkcích spolu s příkladem jejich použití jsou uvedeny v části Rozhraní API zabezpečení dokumentace k produktu IBM i*Application Programming Interface* .
  2. Nezrušujte důvěryhodné aplikace pomocí volby 2 systémového požadavku nebo ukončením úloh, ve kterých jsou spuštěny pomocí ENDJOB.
- ALW V systémech AIX, Linux, and Windows nejsou podporovány důvěryhodné 32bitové aplikace. Pokud se pokusíte spustit důvěryhodnou 32bitovou aplikaci, bude snížena úroveň důvěryhodného připojení na standardní vázané připojení.

Sdílená (nezávislá na vláknu) připojení s MQCONNX
Tyto informace slouží k získání informací o sdílených připojeních pomocí MQCONNX a některých poznámkách k použití, které je třeba zvážit.

Poznámka: Není podporováno na systému IBM MQ for z/OS.

Na jiných platformách IBM MQ než IBM MQ for z/OSje připojení s produktem `MQCONN` k dispozici pouze pro podproces, který připojení vytvořil. Volby volání MQCONNX umožňují vytvořit připojení, které mohou

sdílet všechny podprocesy v procesu. Pokud je vaše aplikace spuštěna v transakčním prostředí, které vyžaduje zadání volání MQI ve stejném podprocesu, musíte použít následující výchozí volbu:

MQCNO_HANDLE_SHARE_NONE
Vytvoří nesdílené připojení.

Ve většině ostatních prostředí můžete použít jednu z následujících voleb sdílených připojení nezávislých na podprocesu:

MQCNO_HANDLE_SHARE_BLOCK
Vytvoří sdílené připojení. V případě připojení MQCNO_HANDLE_SHARE_BLOCK , pokud je připojení aktuálně používáno voláním MQI v jiném podprocesu, bude volání MQI čekat na dokončení aktuálního volání MQI.

MQCNO_HANDLE_SHARE_NO_BLOCK
Vytvoří sdílené připojení. V případě připojení MQCNO_HANDLE_SHARE_NO_BLOCK , pokud je připojení aktuálně používáno voláním MQI v jiném podprocesu, volání MQI okamžitě selže s příčinou MQRC_CALL_IN_PROGRESS.

S výjimkou prostředí MTS (Microsoft Transaction Server) je výchozí hodnota MQCNO_HANDLE_SHARE_NONE. V prostředí MTS je výchozí hodnota MQCNO_HANDLE_SHARE_BLOCK.

Z volání MQCONNX se vrátí manipulátor připojení. Manipulátor může být použit následnými voláními MQI z libovolného podprocesu v procesu, která přidruží tato volání k manipulátoru vráceném z MQCONNX. Volání MQI s použitím jednoho sdíleného manipulátoru jsou serializována napříč podprocesy.

Například následující posloupnost aktivity je možná se sdíleným popisovačem:

1. Podproces 1 vydá příkaz MQCONNX a získá 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 s podprocesem 2 MQDISC pomocí *h1*

Zatímco manipulátor je používán libovolným podprocesem, přístup k připojení není k dispozici ostatním podprocesům. V případech, kdy je přijatelné, aby podproces čekal na dokončení předchozího volání z jiného podprocesu, použijte volbu MQCONNX s volbou MQCNO_HANDLE_SHARE_BLOCK.

Blokování však může způsobit potíže. Předpokládejme, že v kroku “2” na stránce 717podproces 1 vydá požadavek na získání, který čeká na zprávy, které ještě nedorazily (příkaz get with wait). V tomto případě jsou vlákna 2 a 3 také ponechána čekající (blokovaná) tak dlouho, jak trvá požadavek na získání na vlákně 1. Dáváte-li přednost tomu, aby se volání MQI vrátilo s chybou, pokud je na popisovači již spuštěno jiné volání MQI, použijte volbu MQCONNX s volbou MQCNO_HANDLE_SHARE_NO_BLOCK.

Poznámky k použití sdíleného připojení

1. Všechny manipulátory objektů (Hobj) vytvořené otevřením objektu jsou spojeny s Hconnem; takže pro sdílený Hconn jsou Hobjs také sdíleny a použitelné libovolným podprocesem používajícím Hconn. Podobně je jakákoli pracovní jednotka spuštěná v rámci připojení Hconn přidružena k tomuto připojení Hconn; takže i toto je sdíleno mezi podprocesy se sdíleným připojení Hconn.
2. *Jakýkoli* podproces může volat MQDISC k odpojení sdíleného připojení Hconn, nikoli pouze podprocesu, který volal odpovídající MQCONNX. Příkaz MQDISC ukončí objekt Hconn, čímž jej znepřístupní pro všechny podprocesy.
3. Jeden podproces může sériově používat více sdílených připojení Hconns, například pomocí příkazu MQPUT vložit jednu zprávu do jednoho sdíleného připojení Hconn a poté vložit jinou zprávu pomocí jiného sdíleného připojení Hconn, přičemž každá operace je pod jinou lokální pracovní jednotkou.
4. Sdílené konzoly Hconns nelze použít v rámci globální pracovní jednotky.

Multi *Použití voleb volání MQCONNX s MQ_CONNECT_TYPE*

Pomocí těchto informací můžete porozumět různým volbám volání MQCONNX a způsobům jejich použití s proměnnou prostředí **MQ_CONNECT_TYPE** .

Poznámka: Parametr **MQ_CONNECT_TYPE** má vliv pouze na vazby typu STANDARD. Pro ostatní vazby je parametr **MQ_CONNECT_TYPE** ignorován.

V systému IBM MQ for Multiplatformsmůžete použít proměnnou prostředí **MQ_CONNECT_TYPE** v kombinaci s typem vazby zadaným v poli `Options` struktury MQCNO použité ve volání MQCONNX.

*Tabulka 113. Způsob použití voleb volání MQCONNX s proměnnou prostředí **MQ_CONNECT_TYPE***

Volba volání MQCONNX	proměnná prostředí MQ_CONNECT_TYPE	Výsledek
STANDARD	Nedefinovaný	STANDARD
STANDARD	STANDARD	STANDARD
STANDARD	Rychlý	STANDARD
STANDARD	CLIENT	CLIENT
STANDARD	LOKÁLNÍ	STANDARD

Není-li zadána volba MQCNO_STANDARD_BINDING, můžete použít MQCNO_NONE, která má výchozí hodnotu MQCNO_STANDARD_BINDING.

Ověření a identita pro MQCONN a MQCONNX

Pomocí této úlohy zjistíte, jak mohou aplikace dodat pověření, která se používají pro ověření, když se připojují k produktu IBM MQ.

Výchozí identita uživatele

Pokud aplikace používá rozhraní fronty zpráv (MQI) pro připojení k produktu IBM MQ pomocí MQCONN nebo MQCONNX, je identita uživatele vždy vytvořena a přidružena k připojení.

Při výchozím nastavení je počáteční identita uživatele vždy totožná s identitou procesu operačního systému, pod kterým je aplikace spuštěna. Tato počáteční identita může být dostatečná pro lokálně vázaná nebo důvěryhodná připojení aplikací.

Když se aplikace připojí ke správci front pomocí volání MQCONN, nemůže upravit výchozí ID uživatele. Následující mechanizmy však mohou změnit ID uživatele, které je přidruženo k připojení:

- Uživatelská procedura zabezpečení na straně klienta nebo serveru.
- Pravidla ověřování kanálu ve správci front.
- ID uživatele klienta zavedené během vzájemného ověření TLS.

Použití MQCONNX k dodání pověření

MQCONNX poskytuje aplikaci větší kontrolu nad identitou, která je přidružena k připojení. Aplikace může dodat strukturu MQCSP jako součást voleb připojení, které jsou určeny v parametru **ConnectOpts** pro MQCONNX. Struktura MQCSP může obsahovat pověření, která se používají k vytvoření identity uživatele. Produkt IBM MQ podporuje následující pověření ve struktuře MQCSP:

- ID uživatele a heslo.
- V 9.3.4 V produktu IBM MQ 9.3.4se jedná o token ověření, pokud se aplikace připojuje ke správci front, který je spuštěn v systémech AIX nebo Linux .

Ověřování připojení správce front a konfigurace ověřování kanálu řídí způsob zpracování pověření dodaných aplikací. Tato konfigurace například ovlivňuje následující aspekty:

- Zda jsou ověřena pověření ve struktuře MQCSP a jak jsou ověřena.
- Zda je ID uživatele v pověřeních ve struktuře MQCSP mapováno na jiné ID uživatele.
- Určuje, zda je ověřený uživatel převzata jako kontext pro aplikaci.

Další informace o ověřování připojení naleznete v tématu Ověřování připojení. Další informace o ověřování kanálu naleznete v tématu Záznamy ověřování kanálu.

Několik ukázkových programů napsaných v jazyce C, které používají rozhraní MQI, demonstruje použití struktury MQCSP k poskytnutí ověřovacích pověření. Další informace naleznete v následujících ukázkových programech:

- “Ukázkové programy Get” na stránce 1055
- “Ukázkové programy Put” na stránce 1067
- “Ukázkový program prohlížeče” na stránce 1043
- “Ukázkový program TLS” na stránce 1082

Související informace

Identifikace a ověřování uživatelů pomocí struktury MQCSP

MQCSP-parametry zabezpečení

Identifikace a ověřování uživatelů

Odpojení programů od správce front pomocí MQDISC

Pomocí těchto informací získáte informace o odpojení programů od správce front pomocí MQDISC.

Pokud program, který se připojil ke správci front pomocí volání MQCONN nebo MQCONNX, dokončil veškerou interakci se správcem front, přeruší připojení pomocí volání MQDISC, s výjimkou následujících:

- V aplikacích CICS Transaction Server for z/OS , kde je volání volitelné, pokud nebylo použito MQCONNX a chcete zrušit značku připojení před ukončením aplikace.
- V systému IBM MQ for IBM i , kde se při odhlášení od operačního systému provede implicitní volání MQDISC.

Jako vstup volání MQDISC musíte zadat manipulátor připojení (Hconn), který byl vrácen MQCONN nebo MQCONNX při připojení ke správci front.

Po volání MQDISC manipulátor připojení (Hconn) již s výjimkou systému CICS on z/OSnení platný a nelze zadat žádná další volání MQI, dokud znovu nezavoláte MQCONN nebo MQCONNX. MQDISC provádí implicitní MQCLOSE pro všechny objekty, které jsou stále otevřené pomocí tohoto popisovače.

z/OS Pro klienta připojeného k produktu z/OSplatí, že když je vyvoláno volání MQDISC, dojde k implicitnímu potvrzení, ale všechny popisovače front, které jsou stále otevřené, nejsou zavřené, dokud kanál skutečně neskončí.

Použijete-li pro připojení k systému IBM MQ for z/OSpříkaz MQCONNX, příkaz MQDISC také ukončí obor značky připojení vytvořené příkazem MQCONNX. Pokud však v aplikaci CICS, IMSnebo RRS existuje aktivní jednotka zotavení přidružená ke značce připojení, je MQDISC odmítnut s kódem příčiny MQRC_CONN_TAG_NOT_RELEASED.

Popisy parametrů jsou uvedeny v popisu volání MQDISC v adresáři MQDISC.

Není-li zadán příkaz MQDISC

Standardní nesdílené připojení (Hconn) se vyčistí, když se podproces vytváření ukončí. Sdílené připojení je implicitně zálohováno a odpojeno pouze při ukončení celého procesu. Pokud podproces, který vytvořil sdílený adaptér Hconn, bude ukončen, dokud bude připojení Hconn stále existovat, bude připojení Hconn stále použitelné.

Kontrola oprávnění

Volání MQCLOSE a MQDISC obvykle neprovádějí žádnou kontrolu oprávnění.

V normálním průběhu událostí se úloha, která má oprávnění k otevření nebo připojení k objektu IBM MQ , zavře nebo odpojí od tohoto objektu. I když je oprávnění úlohy, která se připojila k objektu IBM MQ nebo jej otevřela, odvoláno, jsou přijata volání MQCLOSE a MQDISC.

Otevírání a zavírání objektů

Tyto informace poskytují náhled na otevírání a zavírání objektů IBM MQ .

Chcete-li provést některou z následujících operací, musíte nejprve *otevřít* příslušný objekt IBM MQ :

- Vložení zpráv do fronty
- Získat (procházet nebo načíst) zprávy z fronty
- Nastavení atributů objektu
- Dotaz na atributy libovolného objektu

Pomocí volání MQOPEN otevřete objekt a pomocí voleb volání určete, co chcete s objektem provést. Jedinou výjimkou je, pokud chcete vložit jednu zprávu do fronty, pak okamžitě zavřete frontu. V tomto případě můžete vynechat fázi *otevírání* pomocí volání MQPUT1 (viz “Vložení jedné zprávy do fronty pomocí volání MQPUT1” na stránce 738).

Před otevřením objektu pomocí volání MQOPEN je nutné připojit program ke správci front. Toto je podrobně vysvětleno pro všechna prostředí v souboru “Připojení ke správci front a odpojení od něj” na stránce 713.

Existují čtyři typy objektů IBM MQ , které můžete otevřít:

- Fronta
- Seznam názvů
- Definice procesu
- Správce front

Všechny tyto objekty otevřete 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 popisovače a odebrat zprávy ze stejné fronty pomocí jiného popisovače. Tím se ušetří použití prostředků k zavření a opětovnému otevření stejného objektu. Můžete také otevřít frontu pro procházení *a* odebrání zpráv současně.

Kromě toho můžete otevřít více objektů s jedním objektem MQOPEN a zavřít je pomocí příkazu MQCLOSE. Informace o tom, jak to provést, naleznete v části “Distribuční seznamy” na stránce 739 .

Při pokusu o otevření objektu správce front zkontroluje, zda máte oprávnění k otevření tohoto objektu pro volby zadané ve volání MQOPEN.

Objekty se zavřou automaticky, když se program odpojí od správce front. V prostředí IMS je odpojení vynuceno, když program zahájí zpracování pro nového uživatele po volání GU (get unique) IMS . Na platformě IBM i se objekty zavírají automaticky po ukončení úlohy.

Je dobrým zvykem zavírat objekty, které jste otevřeli. K tomu použijte volání MQCLOSE.

Pomocí následujících odkazů se dozvíte více o otevírání a zavírání objektů:

- “Otevírání objektů pomocí volání MQOPEN” na stránce 721
- “Vytváření dynamických front” na stránce 728
- “Otevírání vzdálených front” na stránce 729
- “Zavření objektů pomocí volání MQCLOSE” na stránce 729

Související pojmy

“Přehled rozhraní fronty zpráv” na stránce 700
Informace o komponentách rozhraní MQI (Message Queue Interface).

“Připojení ke správci front a odpojení od něj” na stránce 713

Chcete-li používat programovací služby IBM MQ , musí mít program připojení ke správci front. Pomocí těchto informací se dozvíte, jak se připojit ke správci front a jak se od něj odpojit.

"Vkládání zpráv do fronty" na stránce 730
Pomocí těchto informací se dozvíte, jak vkládat zprávy do fronty.

"Získávání zpráv z fronty" na stránce 744
Pomocí těchto informací získáte informace o získávání zpráv z fronty.

"Zjišťování a nastavení atributů objektu" na stránce 822
Atributy jsou vlastnosti, které definují charakteristiky objektu IBM MQ .

"Potvrzení a zálohování jednotek práce" na stránce 825
Tyto informace popisují, jak potvrdit a vrátit zpět všechny obnovitelné operace get a put, které se vyskytly v transakci.

"Spuštění aplikací IBM MQ pomocí spouštěčů" na stránce 836
Získejte informace o spouštěčích a o tom, jak spouštět aplikace IBM MQ pomocí spouštěčů.

"Práce s rozhraním MQI a klastry" na stránce 854
Existují speciální volby pro volání a návratové kódy, které se týkají klastrování.

"Používání a psaní aplikací na systému IBM MQ for z/OS" na stránce 858
Aplikace IBM MQ for z/OS lze vytvořit z programů, které jsou spuštěny v mnoha různých prostředích. To znamená, že mohou využívat zařízení, která jsou k dispozici ve více než jednom prostředí.

"IMS a IMS přemostění aplikací na IBM MQ for z/OS" na stránce 66
Tyto informace vám pomohou psát aplikace IMS pomocí IBM MQ.

Otevírání objektů pomocí volání MQOPEN

Pomocí těchto informací získáte informace o otevírání objektů pomocí volání MQOPEN.

Jako vstup pro volání MQOPEN musíte zadat:

- Manipulátor připojení. Pro aplikace systému CICS v systému z/OSmůžete určit konstantu MQHC_DEF_HCONN (která má hodnotu nula) nebo použít manipulátor připojení vrácený voláním MQCONN nebo MQCONNX. Pro ostatní programy vždy použijte manipulátor připojení vrácený voláním MQCONN nebo MQCONNX.
- Popis objektu, který chcete otevřít, pomocí struktury deskriptoru objektu (MQOD).
- Jedna nebo více voleb, které řídí akci volání.

Výstup z MQOPEN je:

- Popisovač objektu, který představuje váš přístup k objektu. Tuto volbu použijte na vstupu pro všechna následná volání MQI.
- Modifikovaná struktura deskriptoru objektu, pokud vytváříte dynamickou frontu (a je podporována na vaší platformě).
- Kód dokončení.
- Kód příčiny.

Rozsah popisovače objektu

Rozsah popisovače objektu (Hobj) je stejný jako rozsah popisovače připojení (Hconn).

Tato položka je uvedena v části "Rozsah MQCONN nebo MQCONNX" na stránce 715 a "Sdílená (nezávislá na vláknu) připojení s MQCONNX" na stránce 716. V některých prostředích však existují další aspekty:

CICS
V programu CICS můžete manipulátor použít pouze v rámci stejné úlohy CICS , ze které jste provedli volání MQOPEN.

IMS a z/OS dávka
V prostředí IMS a v dávkovém prostředí můžete použít popisovač v rámci stejné úlohy, ale ne v rámci žádných dílčích úloh.

Popisy parametrů volání MQOPEN jsou uvedeny v části MQOPEN.

Následující sekce popisují informace, které musíte zadat jako vstup pro MQOPEN.

Identifikace objektů (struktura MQOD)

Pomocí struktury MQOD identifikujte objekt, který chcete otevřít. Tato struktura je vstupním parametrem pro volání MQOPEN. (Struktura je upravena správcem front při použití volání MQOPEN k vytvoření dynamické fronty.)

Úplné podrobnosti o struktuře MQOD viz MQOD.

Informace o použití struktury MQOD pro distribuční seznamy viz “Použití struktury MQOD” na stránce 740 v části “Distribuční seznamy” na stránce 739.

Rozlišení názvu
Způsob, jakým volání MQOPEN interpretuje 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 překladu názvů pro zadaný název fronty. Tato volba určuje, ve které frontě má správce front provádět následné operace. To znamená, že když zadáte název alias fronty nebo vzdálené fronty v deskriptoru objektu (MQOD), volání přeloží název buď na lokální frontu, nebo na přenosovou frontu. Je-li fronta otevřena pro jakýkoli typ vstupu, procházení nebo nastavení, interpretuje se jako lokální fronta, pokud existuje, a selže, pokud neexistuje. Je vyhodnocována jako nelokální fronta pouze v případě, že je otevřena pouze pro výstup, pouze pro dotazování nebo pouze pro výstup a dotazování. Přehled procesu rozpoznávání názvů naleznete v části Tabulka 114 na stránce 722 . Název, který zadáte v poli *ObjectQMgrName* , je vyřešen *před* názvem v souboru *ObjectName*.

Tabulka 114 na stránce 722 také ukazuje, jak můžete použít lokální definici vzdálené fronty k definování aliasu pro název správce front. To vám umožňuje vybrat, která přenosová fronta se použije při vložení zpráv do vzdálené fronty, takže můžete 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 pro MQOD**a vyberte příslušný případ. Pak si přečtěte přes odpovídající řádek a postupujte podle pokynů. Podle pokynů ve sloupcích **Vyřešené názvy** se můžete buď vrátit do sloupců **Vstup pro MQOD** a vložit hodnoty podle pokynů, nebo můžete ukončit tabulku s dodanými výsledky. Můžete být například požádáni o zadání hodnoty *ObjectName*.

Tabulka 114. Řešení názvů front při použití MQOPEN

Vstup pro MQOD	**Vstup pro MQOD**	**Rozlišené názvy**	**Rozlišené názvy**	**Rozlišené názvy**
ObjectQMgrName	*ObjectName*	*ObjectQMgrName*	*ObjectName*	Transmission queue
Prázdný nebo lokální správce front	Lokální fronta bez atributu CLUSTER	Lokální správce front	Vstup *ObjectName*	Nelze použít (použita lokální fronta)
Prázdný správce front	Lokální fronta s atributem CLUSTER	Vybraný správce front klastru správy pracovní zátěže nebo specifický správce front klastru vybraný v PUT	Vstup *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	Vstup *ObjectName*	Nelze použít (použita lokální fronta)

Tabulka 114. Řešení názvů front při použití MQOPEN (pokračování)

Vstup pro MQOD	**Vstup pro MQOD**	**Rozlišené názvy**	**Rozlišené názvy**	**Rozlišené názvy**
Prázdný nebo lokální správce front	Modelová fronta	Lokální správce front	Generovaný název	Nelze použít (použita lokální fronta)
Prázdný nebo lokální správce front	Alias fronty s nebo bez atributu CLUSTER	Znovu proveďte rozpoznání názvu s nezměněným názvem *ObjectQMgrName* a zadejte *ObjectName* nastavený na hodnotu *BaseQName* v objektu definice alias fronty. Nesmí se interpretovat jako alias lokálně definovaný, kde je uveden *ObjectQMgrName* , ale může se interpretovat jako klastrovaný alias (hostovaný na jiných správcích front), kde je *ObjectQMgrName* prázdný.		
Lokální správce front	Alias fronty s atributem CLUSTER	Alias se nesmí interpretovat jako fronta klastru, která není lokálně definována, nebo jako fronta klastru, která má stejný *ObjectName* jako alias.		
Prázdný správce front	Alias fronty s atributem CLUSTER	Alias se může interpretovat jako fronta klastru se stejným *ObjectName* jako alias.		
Prázdný nebo lokální správce front	lokální definice vzdálené fronty	Znovu proveďte rozpoznání názvu s parametrem *ObjectQMgrName* nastaveným na *RemoteQMgrName*a *Obj ectName* nastaveným na *RemoteQName*. Vzdálené fronty nesmí být vyřešeny.		Název atributu *XmitQName* , pokud není mezerové; jinak *RemoteQMgrNázev* v objektu definice vzdálené fronty. SYSTEM.QSG.TRANSMI T.QUEUE (viz poznámka)
Prázdný správce front	Nebyl nalezen žádný odpovídající lokální objekt; fronta klastru	Vybraný správce front klastru správy pracovní zátěže nebo specifický správce front klastru vybraný v PUT	Vstup *ObjectName*	SYSTEM.CLUSTER.TRAN SMIT.QUEUE SYSTEM.QSG.TRANSMI T.QUEUE (viz poznámka)

Tabulka 114. Řešení názvů front při použití MQOPEN (pokračování)				
Vstup pro MQOD	**Vstup pro MQOD**	**Rozlišené názvy**	**Rozlišené názvy**	**Rozlišené názvy**
Prázdný nebo lokální správce front	Žádný 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	Vstup *ObjectName*	Nelze použít
Název lokální přenosové fronty	(Nevyřešeno)	Název vstupu *ObjectQMgr*	Vstup *ObjectName*	Název vstupu *ObjectQMgr* SYSTEM.QSG.TRANSMIT.QUEUE (viz poznámka)
Definice aliasu správce front (*RemoteQMgrNázev* může být lokální správce front)	(Nevyřešeno, vzdálená fronta)	Znovu proveďte rozpoznávání názvů s volbou *ObjectQMgrName* nastavenou na hodnotu *RemoteQMgrName*. Nesmí se interpretovat jako vzdálené fronty	Vstup *ObjectName*	Název atributu *XmitQName* , pokud není mezerové; jinak *RemoteQMgrNázev* v objektu definice vzdálené fronty. SYSTEM.QSG.TRANSMIT.QUEUE (viz poznámka)
Správce front není názvem žádného lokálního objektu. Nalezeni jsou správci front klastru nebo alias správce front.	(Nevyřešeno)	*ObjectQMgrNázev* nebo specifický správce front klastru vybraný v PUT	Vstup *ObjectName*	SYSTEM.CLUSTER.TRANSMIT.QUEUE SYSTEM.QSG.TRANSMIT.QUEUE (viz poznámka)
Správce front není názvem žádného lokálního objektu; nebyly nalezeny žádné objekty klastru	(Nevyřešeno)	Název vstupu *ObjectQMgr*	Vstup *ObjectName*	*DefXmitQName* atribut správce front, kde 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 IBM MQ for z/OS , kteří jsou součástí skupiny sdílení front (QSG), lze v produktu Tabulka 114 na stránce 722použít název skupiny sdílení front namísto názvu lokálního správce front.

Pokud lokální správce front nemůže otevřít cílovou frontu nebo vložit zprávu do fronty, bude zpráva přenesena do určeného názvu ObjectQMgrprostřednictvím fronty v rámci skupiny nebo kanálu IBM MQ .

6. Ve sloupci *ObjectName* tabulky odkazuje CLUSTER na atributy CLUSTER a CLUSNL fronty.
7. Systém SYSTEM.QSG.TRANSMIT.QUEUE se používá, pokud jsou lokální a vzdálení správci front ve stejné skupině sdílení front; je povoleno řazení do front v rámci skupiny.
8. Pokud jste ke každému odesílacímu kanálu klastru přiřadili jinou přenosovou frontu klastru, SYSTEM.CLUSTER.TRANSMIT.QUEUE nemusí být název přenosové fronty klastru. Další informace o více přenosových frontách klastru naleznete v tématu Klastrování: Plánování konfigurace přenosových front klastru.
9. V situaci, kdy správce front není názvem žádného lokálního objektu, nalezeného správce front klastru nebo aliasu správce front.

   Pokud jste zadali název správce front pomocí produktu **ObjectQMgrName**a existuje více kanálů klastru s různými názvy klastrů, které lokální správce front zná a které by dosáhly tohoto cíle, může být kterýkoli z těchto kanálů použit k přesunutí zprávy bez ohledu na název klastru cílové fronty.

   To může být neočekávané, pokud jste očekávali, že zprávy pro tuto frontu budou odesílány pouze prostřednictvím kanálu, který má stejný název klastru jako fronta.

   V tomto případě však má přednost konzola **ObjectQMgrName** a vyvážení pracovní zátěže klastru bere v úvahu všechny kanály, které se mohou dostat k tomuto správci front, bez ohledu na název klastru, v němž se nacházejí.

Otevření alias fronty také otevře základní frontu, na kterou se alias interpretuje, a otevření vzdálené fronty také otevře přenosovou frontu. Proto nemůžete odstranit buď frontu, kterou jste uvedli, nebo frontu, do které se vyřeší, když je druhá otevřená.

Zatímco alias frontu nelze interpretovat na jinou lokálně definovanou alias frontu (sdílenou v klastru či nikoli), interpretace na vzdáleně definovanou alias frontu klastru je povolena, a proto ji lze zadat jako základní frontu.

Vyřešený název fronty a vyřešený název správce front jsou uloženy v polích *ResolvedQName* a *ResolvedQMgrName* v produktu MQOD.

Další informace o rozlišování názvů v prostředí distribuovaných front naleznete v tématu Co je rozlišování názvů front?.

Použití voleb volání MQOPEN

V parametru **Options** volání MQOPEN musíte vybrat jednu nebo více voleb pro řízení přístupu k objektu, který otevíráte. Pomocí těchto voleb můžete:

- Otevřít frontu a určit, že všechny zprávy vkládané do této fronty musí být směrovány na stejnou její instanci
- Otevřete frontu, abyste do ní mohli vkládat zprávy.
- Otevřete frontu, abyste v ní mohli procházet zprávy.
- Otevřete frontu, abyste z ní mohli odebrat zprávy.
- Otevřete objekt, abyste se mohli dotazovat a nastavovat jeho atributy (ale můžete nastavit pouze atributy front)
- Otevřít téma nebo řetězec tématu a publikovat do něj zprávy
- Přidružit informace o kontextu ke zprávě
- Určit alternativní identifikátor uživatele, který se má použít pro kontroly zabezpečení
- Řídit volání, pokud je správce front ve stavu uvedení do klidového stavu

Volba MQOPEN pro frontu klastru
Vazba použitá pro manipulátor fronty je převzata z atributu fronty **DefBind** , který může mít hodnotu MQBND_BIND_ON_OPEN, MQBND_BIND_NOT_FIXEDnebo MQBND_BIND_ON_GROUP.

Chcete-li směrovat všechny zprávy vkládané do fronty pomocí produktu MQPUT do stejného správce front pomocí stejné trasy, použijte volbu MQOO_BIND_ON_OPEN ve volání MQOPEN .

Chcete-li určit, že místo určení má být vybráno v době MQPUT, tj. v jednotlivých zprávách, použijte volbu MQOO_BIND_NOT_FIXED pro volání MQOPEN .

Chcete-li určit, že všechny zprávy ve skupinách zpráv vkládané do fronty pomocí MQPUT jsou přiděleny stejné cílové instanci, použijte volbu MQOO_BIND_ON_GROUP ve volání MQOPEN .

Buď MQOO_BIND_ON_OPEN , nebo MQOO_BIND_ON_GROUP musí být zadáno při použití skupin zpráv s klastry, aby se zajistilo, že všechny zprávy ve skupině budou zpracovány ve stejném místě určení.

Pokud nezadáte žádnou z těchto voleb, použije se výchozí hodnota MQOO_BIND_AS_Q_DEF.

Zadáte-li název správce front v souboru MQOD, bude vybrána fronta v tomto správci front. Je-li název správce front prázdný, lze vybrat libovolnou instanci. Další informace viz “MQOPEN a klastry” na stránce 855.

Otevřete-li frontu klastru pomocí definice QALIAS , některé atributy fronty jsou definovány alias frontou, nikoli základní frontou. Atributy klastru patří mezi atributy definice základní fronty, které jsou přepsány alias frontou. Například v následujícím úseku kódu se fronta klastru otevře s MQOO_BIND_NOT_FIXED a ne MQOO_BIND_ON_OPEN. Definice fronty klastru je v rámci klastru inzerována, definice alias fronty je pro správce front lokální.

```
DEFINE QLOCAL(CLQ1) CLUSTER(MYCLUSTER) DEFBIND(OPEN) REPLACE
DEFINE QALIAS(ACLQ1) TARGET(CLQ1) DEFBIND(NOTFIXED) REPLACE
```

Volba MQOPEN pro vkládání zpráv
Chcete-li otevřít frontu nebo téma a vložit do ní zprávy, použijte volbu MQOO_OUTPUT.

Volba MQOPEN pro procházení zpráv
Chcete-li otevřít frontu, abyste mohli *procházet* zprávy v ní uvedené, 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 776.

Poznámka:

1. Zprávy ve vzdálené frontě nelze procházet; neotvírejte vzdálenou frontu pomocí volby MQOO_BROWSE.
2. Tuto volbu nemůžete uvést při otevírání rozdělovníku. Další informace o distribučních seznamech viz “Distribuční seznamy” na stránce 739.
3. Použijte MQOO_CO_OP ve spojení s MQOO_BROWSE, pokud používáte kooperativní procházení; viz Volby

Volby MQOPEN pro odebrání zpráv
Otevření fronty pro odebrání zpráv z ní řídí tři možnosti.

V libovolném volání MQOPEN můžete použít pouze jeden z nich. Tyto volby definují, zda má váš program výlučný nebo sdílený přístup k frontě. *Výlučný přístup* znamená, že dokud nezavřete frontu, pouze vy z něj můžete odebrat zprávy. Pokud se jiný program pokusí otevřít frontu a odebrat zprávy, jeho volání MQOPEN se nezdaří. *Sdílený přístup* znamená, že lze odebrat více než jeden program. zprávy z fronty.

Nejvhodnějším přístupem je přijmout typ přístupu, který byl určen pro frontu při definování fronty. Definice fronty zahrnovala nastavení **Shareability** a **DefInputOpenOption** atributů. Chcete-li tento přístup přijmout, použijte volbu MQOO_INPUT_AS_Q_DEF. Informace o tom, jak nastavení těchto atributů ovlivňuje typ přístupu, který vám bude poskytnut při použití této volby, naleznete v části Tabulka 115 na stránce 727 .

Tabulka 115. Jak atributy fronty a volby volání MQOPEN ovlivňují 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
NESDÍLENÉ_POLOŽKY *	SHARED*	výlučný	výlučný	výlučný
NESDÍLENÉ_SDÍLENÍ	EXCLUSIVE	výlučný	výlučný	výlučný

Poznámka: * Ačkoli můžete definovat frontu tak, aby měla tuto kombinaci atributů, výchozí volba otevření vstupu je přepsána atributem možnosti sdílení.

Alternativně:

- Pokud víte, že vaše aplikace může úspěšně pracovat i v případě, že jiné programy mohou současně odebírat zprávy z fronty, použijte volbu MQOO_INPUT_SHARED. Tabulka 115 na stránce 727 ukazuje, jak vám v některých případech bude udělen výlučný přístup k frontě, a to i s touto volbou.
- Pokud víte, že vaše aplikace může úspěšně pracovat pouze v případě, že je současně zabráněno ostatním programům v odebrání zpráv z fronty, použijte volbu MQOO_INPUT_EXCLUSIVE.

Poznámka:

1. Nelze odebrat zprávy ze vzdálené fronty. Proto nemůžete otevřít vzdálenou frontu pomocí žádné z voleb MQOO_INPUT_ *.
2. Tuto volbu nemůžete uvést při otevírání rozdělovníku. Další informace uvádí téma "Distribuční seznamy" na stránce 739.

Volby MQOPEN pro nastavení a dotazování na atributy
Chcete-li otevřít frontu, abyste mohli nastavit její atributy, použijte volbu MQOO_SET.

Nelze nastavit atributy žádného jiného typu objektu (viz "Zjišťování a nastavení atributů objektu" na stránce 822).

Chcete-li otevřít objekt, abyste se mohli dotázat na jeho atributy, použijte volbu MQOO_INQUIRE.

Poznámka: Tuto volbu nemůžete uvést při otevírání rozdělovníku.

Volby MQOPEN týkající se kontextu zprávy
Chcete-li mít možnost přidružit informace o kontextu ke zprávě při jejím vložení do fronty, musíte při otevření fronty použít jednu z voleb kontextu zprávy.

Volby vám umožňují rozlišovat mezi informacemi o kontextu, které se vztahují k *uživateli* , který zprávu vytvořil, a informacemi, které se vztahují k *aplikaci* , z níž zpráva pochází. Můžete se také rozhodnout nastavit informace o kontextu při vložení zprávy do fronty, nebo můžete zvolit, aby byl kontext automaticky převzat z jiného manipulátoru fronty.

Související pojmy
"kontext zprávy" na stránce 45
Kontext zprávy umožňuje aplikaci, která načte zprávu, zjistit informace o původci zprávy.

"Řízení informací o kontextu zprávy" na stránce 736
Při použití volání MQPUT nebo MQPUT1 k vložení zprávy do fronty můžete určit, že správce front má do deskriptoru zprávy přidat některé výchozí informace o kontextu. Aplikace, které mají odpovídající úroveň oprávnění, mohou přidat další informace o kontextu. Pole voleb ve struktuře MQPMO můžete použít k řízení informací o kontextu.

Volba MQOPEN pro alternativní oprávnění uživatele
Při pokusu o otevření objektu pomocí volání MQOPEN správce front zkontroluje, zda máte oprávnění k otevření tohoto objektu. Pokud nemáte oprávnění, volání selže.

Programy serveru však mohou požadovat, aby správce front zkontrolovali autorizaci uživatele, pro kterého pracují, a nikoli vlastní autorizaci serveru. K tomu musí použít volbu MQOO_ALTERNATE_USER_AUTHORITY volání MQOPEN a zadat alternativní ID uživatele do pole *AlternateUserId* struktury MQOD. Server obvykle získá ID uživatele z informací o kontextu ve zprávě, kterou zpracovává.

z/OS *Volba MQOPEN pro uvedení správce front do klidového stavu*

Pokud použijete volání MQOPEN, když je správce front ve stavu uvedení do klidového stavu, může volání v závislosti na používaném prostředí selhat.

Pokud v prostředí CICS v systému z/OSpoužijete volání MQOPEN, když je správce front v klidovém stavu, volání vždy selže.

V jiných prostředích z/OS a Multiplatforms se volání nezdaří, pokud je správce front uveden do klidového stavu pouze v případě, že používáte volbu MQOO_FAIL_IF_QUIESCING volání MQOPEN.

Volba MQOPEN pro překlad názvů lokálních front

Když otevřete lokální, alias nebo modelovou frontu, vrátí se lokální fronta.

Při otevření vzdálené fronty nebo fronty klastru jsou však pole *ResolvedQName* a *ResolvedQMgrName* struktury MQOD vyplněna názvy vzdálené fronty a vzdáleného správce front nalezeného v definici vzdálené fronty nebo vybranou vzdálenou frontou klastru.

Pomocí volby MQOO_RESOLVE_LOCAL_Q volání MQOPEN vyplňte *ResolvedQName* ve struktuře MQOD názvem otevřené lokální fronty. *ResolvedQMgrName* je podobně vyplněn názvem lokálního správce front, který je hostitelem lokální fronty. Toto pole je k dispozici pouze pro verzi 3 struktury MQOD. Je-li struktura nižší než verze 3, bude hodnota MQOO_RESOLVE_LOCAL_Q ignorována bez vrácení chyby.

Zadáte-li při otevírání proměnnou MQOO_RESOLVE_LOCAL_Q, například vzdálenou frontu, bude proměnná *ResolvedQName* představovat název přenosové fronty, do níž budou vkládány zprávy. *ResolvedQMgrName* je název lokálního správce front, který je hostitelem přenosové fronty.

Vytváření dynamických front

Dynamickou frontu použijte v případě, že ji po ukončení aplikace nepotřebujete.

Můžete například použít dynamickou frontu pro svou frontu pro odpovědi. Název fronty pro odpověď zadáváte do pole *ReplyToQ* struktury MQMD při vložení zprávy do fronty (viz “Definování zpráv pomocí struktury MQMD” na stránce 731).

Chcete-li vytvořit dynamickou frontu, použijte šablonu známou jako modelová fronta spolu s voláním MQOPEN. Modelovou frontu vytvoříte pomocí příkazů IBM MQ nebo pomocí operací a ovládacích panelů. Dynamická fronta, kterou vytvoříte, převezme atributy modelové fronty.

Při volání MQOPEN zadejte název modelové fronty do pole *ObjectName* struktury MQOD. Po dokončení volání je pole *ObjectName* nastaveno na název vytvořené dynamické fronty. Pole *ObjectQMgrName* je také nastaveno na název lokálního správce front.

Název dynamické fronty, kterou vytvoříte, můžete zadat třemi způsoby:

- Do pole *DynamicQName* struktury MQOD zadejte požadovaný úplný název.
- Zadejte pro název předponu (méně než 33 znaků) a umožněte správci front generovat zbytek názvu. To znamená, že správce front generuje jedinečný název, ale stále máte určitou kontrolu (například můžete chtít, aby každý uživatel používal určitou předponu, nebo můžete dát speciální klasifikaci zabezpečení frontám s určitou předponou v názvu). Chcete-li použít tuto metodu, uveďte hvězdičku (*) pro poslední neprázdný znak pole *DynamicQName* . Pro název dynamické fronty nezadávejte jedinou hvězdičku (*).
- Povolit správci front generovat úplný název. Chcete-li použít tuto metodu, uveďte hvězdičku (*) na pozici prvního znaku pole *DynamicQName* .

Další informace o těchto metodách naleznete v popisu pole DynamicQName .

Další informace o dynamických frontách naleznete v tématu Dynamické a modelové fronty.

Otevírání vzdálených front

Vzdálená fronta je fronta, kterou vlastní 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 takto:

1. Do pole *ObjectName* struktury MQOD zadejte název vzdálené fronty, jak jej zná *lokální* správce front.

   Poznámka: V tomto případě ponechte pole *ObjectQMgrName* prázdné.
2. Do pole *ObjectName* struktury MQOD zadejte název vzdálené fronty tak, jak jej zná *vzdálený* správce front. Do pole *ObjectQMgrName* zadejte jednu z následujících možností:

   - Název přenosové fronty, která má stejný název jako vzdálený správce front. Název a velikost písmen (velká písmena, malá písmena nebo směs) se musí shodovat *přesně*.
   - Název objektu aliasu správce front, který se interpretuje jako cílový správce front nebo přenosová fronta.

   To sděluje správci front místo určení zprávy a také přenosovou frontu, kterou je třeba vložit, aby se tam dostal.
3. Je-li podporován produkt *DefXmitQname* , zadejte do pole *ObjectName* struktury MQOD název vzdálené fronty tak, jak jej zná *vzdálený* správce front.

   Poznámka: Nastavte pole *ObjectQMgrName* na název vzdáleného správce front (v tomto případě jej nelze ponechat prázdný).

Při volání MQOPEN jsou ověřovány pouze lokální názvy. Poslední kontrolou je existence přenosové fronty, která má být použita.

Tyto metody jsou shrnuty v souboruTabulka 114 na stránce 722.

Zavření objektů pomocí volání MQCLOSE

Chcete-li zavřít objekt, použijte volání MQCLOSE.

Je-li objekt frontou, povšimněte si následujícího:

- Před uzavřením dočasné dynamické fronty není nutné ji vyprázdnit.

  Když zavřete dočasnou dynamickou frontu, fronta se odstraní spolu se všemi zprávami, které v ní mohou být i nadále. To platí i v případě, že pro frontu existují nepotvrzená volání MQGET, MQPUT nebo MQPUT1 .
- Máte-li v systému IBM MQ for z/OSnějaké požadavky MQGET s nevyřízenou volbou MQGMO_SET_SIGNAL pro tuto frontu, budou zrušeny.
- Pokud jste otevřeli frontu pomocí volby MQOO_BROWSE, kurzor procházení bude zničen.

Uzavření nesouvisí s bodem synchronizace, takže můžete zavřít fronty před nebo po bodu synchronizace.

Jako vstup pro volání MQCLOSE musíte dodat:

- Manipulátor připojení. Použijte stejný manipulátor připojení, který jste použili k jeho otevření, nebo případně pro aplikace CICS v systému z/OS, můžete zadat konstantu MQHC_DEF_HCONN (která má hodnotu nula).
- Popisovač objektu, který chcete zavřít. Získejte tuto hodnotu z výstupu volání MQOPEN.
- MQCO_NONE v poli *Options* (pokud nezavíříte trvalou dynamickou frontu).
- Řídicí volba určující, zda má správce front odstranit frontu i v případě, že v ní stále existují zprávy (při zavírání trvalé dynamické fronty).

Výstup příkazu MQCLOSE:

- Kód dokončení
- Kód příčiny
- Popisovač objektu, resetovat na hodnotu MQHO_UNUSABLE_HOBJ

Popisy parametrů volání MQCLOSE jsou uvedeny v tabulce MQCLOSE.

Vkládání zpráv do fronty

Pomocí těchto informací se dozvíte, jak vkládat zprávy do fronty.

K vložení zpráv do fronty použijte volání MQPUT. Pomocí příkazu MQPUT můžete opakovaně vkládat mnoho zpráv do stejné fronty po počátečním volání MQOPEN. Po dokončení vkládání všech zpráv do fronty volejte příkaz MQCLOSE.

Chcete-li vložit jednu zprávu do fronty a zavřít ji ihned poté, můžete použít volání MQPUT1 . MQPUT1 provádí stejné funkce jako následující posloupnost volání:

- MQOPEN
- MQPUT
- MQCLOSE

Obecně však platí, že máte-li pro vložení do fronty více než jednu zprávu, je efektivnější použít volání MQPUT. To závisí na velikosti zprávy a platformě, na které pracujete.

Další informace o vkládání zpráv do fronty naleznete v následujících odkazech:

- “Vložení zpráv do lokální fronty pomocí volání MQPUT” na stránce 731
- “Vkládání zpráv do vzdálené fronty” na stránce 735
- “Nastavení vlastností zprávy” na stránce 735
- “Řízení informací o kontextu zprávy” na stránce 736
- “Vložení jedné zprávy do fronty pomocí volání MQPUT1” na stránce 738
- “Distribuční seznamy” na stránce 739
- “Některé případy, kdy volání put selhávají” na stránce 743

Související pojmy

“Přehled rozhraní fronty zpráv” na stránce 700
Informace o komponentách rozhraní MQI (Message Queue Interface).

“Připojení ke správci front a odpojení od něj” na stránce 713
Chcete-li používat programovací služby IBM MQ , musí mít program připojení ke správci front. Pomocí těchto informací se dozvíte, jak se připojit ke správci front a jak se od něj odpojit.

“Otevírání a zavírání objektů” na stránce 720
Tyto informace poskytují náhled na otevírání a zavírání objektů IBM MQ .

“Získávání zpráv z fronty” na stránce 744
Pomocí těchto informací získáte informace o získávání zpráv z fronty.

“Zjišťování a nastavení atributů objektu” na stránce 822
Atributy jsou vlastnosti, které definují charakteristiky objektu IBM MQ .

“Potvrzení a zálohování jednotek práce” na stránce 825
Tyto informace popisují, jak potvrdit a vrátit zpět všechny obnovitelné operace get a put, které se vyskytly v transakci.

“Spuštění aplikací IBM MQ pomocí spouštěčů” na stránce 836
Získejte informace o spouštěčích a o tom, jak spouštět aplikace IBM MQ pomocí spouštěčů.

“Práce s rozhraním MQI a klastry” na stránce 854
Existují speciální volby pro volání a návratové kódy, které se týkají klastrování.

“Používání a psaní aplikací na systému IBM MQ for z/OS” na stránce 858
Aplikace IBM MQ for z/OS lze vytvořit z programů, které jsou spuštěny v mnoha různých prostředích. To znamená, že mohou využívat zařízení, která jsou k dispozici ve více než jednom prostředí.

“IMS a IMS přemostění aplikací na IBM MQ for z/OS” na stránce 66

Tyto informace vám pomohou psát aplikace IMS pomocí IBM MQ.

Vložení zpráv do lokální fronty pomocí volání MQPUT

Pomocí těchto informací získáte informace o vkládání zpráv do lokální fronty pomocí volání MQPUT.

Jako vstup pro volání MQPUT musíte zadat:

- Manipulátor připojení (Hconn).
- Popisovač fronty (Hobj).
- Popis zprávy, kterou chcete vložit do fronty. Jedná se o strukturu deskriptoru zpráv (MQMD).
- Řídicí informace ve formě struktury voleb vložení zprávy (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, vrátí také strukturu voleb a strukturu deskriptoru zpráv. Volání upraví strukturu voleb tak, aby zobrazovala název fronty a správce front, do kterého byla zpráva odeslána. Pokud požadujete, aby správce front vygeneroval jedinečnou hodnotu pro identifikátor zprávy, kterou vkládáte (zadáním binární nuly do pole *MsgId* struktury MQMD), volání vloží hodnotu do pole *MsgId* před tím, než vám tuto strukturu vrátí. Před zadáním jiného příkazu MQPUT tuto hodnotu resetujte.

V souboru MQPUTje uveden popis volání MQPUT.

Další popis informací potřebných jako vstup pro volání MQPUT naleznete v následujících odkazech:

- “Určení popisovačů” na stránce 731
- “Definování zpráv pomocí struktury MQMD” na stránce 731
- “Určení voleb pomocí struktury MQPMO” na stránce 732
- “Data ve vaší zprávě” na stránce 734
- “Vkládání zpráv: Použití popisovačů zpráv” na stránce 735

Určení popisovačů

Pro manipulátor připojení (*Hconn*) v aplikacích CICS on 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 používejte manipulátor připojení vrácený voláním MQCONN nebo MQCONNX.

Bez ohledu na prostředí, ve kterém pracujete, použijte stejný manipulátor 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 pro zprávu zadána hodnota MQPRI_PRIORITY_AS_AS_Q_DEF nebo MQPER_PERSISTENCE_AS_Q_DEF a fronta je frontou klastru, budou použity hodnoty fronty, pro kterou je MQPUT vyhodnocován. Je-li tato fronta pro příkaz MQPUT zakázána, volání se nezdaří. Další informace naleznete v tématu Konfigurace klastru správců front .

Poznámka: Před vložením nové zprávy použijte MQPMO_NEW_MSG_ID a MQPMO_NEW_CORREL_ID, abyste se ujistili, že jsou *MsgId* a *CorrelId* jedinečné. Hodnoty v těchto polích jsou vráceny v úspěšné MQPUT.

K dispozici je úvod do vlastností zprávy, které modul MQMD popisuje v souboru “Zprávy produktu IBM MQ” na stránce 18, a popis samotné struktury v modulu MQMD.

Určení voleb pomocí struktury MQPMO

Použijte strukturu MQPMO (Put Message Option) k předání voleb do volání MQPUT a MQPUT1 .

Následující sekce vám pomohou s vyplněním polí této struktury. V produktu MQPMOje uveden popis struktury.

Struktura zahrnuje následující pole:

- *StrucId*
- *Version*
- *Options*
- *Context*
- *ResolvedQName*
- *ResolvedQMgrName*
- *RecsPresent*
- *PutMsgRecsFields*
- *ResponseRecOffset and ResponseRecPtr*
- *OriginalMsgHandle*
- *NewMsgHandle*
- *Action*
- *PubLevel*

Obsah těchto polí je následující:

StrucId
Tato volba identifikuje strukturu jako strukturu voleb vložení zprávy. Toto je 4znakové 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 739). Zadáte-li hodnotu MQPMO_VERSION_3, můžete použít manipulátory zpráv a vlastnosti zpráv. Pokud zadáte MQPMO_CURRENT_VERSION, vaše aplikace bude vždy nastavena tak, aby používala nejnovější úroveň.

Volby
To řídí následující:

- Zda je operace vložení zahrnuta do jednotky práce
- Kolik informací o kontextu je přidruženo ke zprávě
- Odkud jsou převzaty informace o kontextu
- Zda se volání nezdaří, pokud je správce front ve stavu uvedení do klidového stavu
- Zda je povoleno seskupování nebo segmentace
- Generování nového identifikátoru zprávy a identifikátoru korelace
- Pořadí, ve kterém jsou zprávy a segmenty vkládány do fronty
- Zda se mají interpretovat názvy lokálních front

Ponecháte-li pole *Options* nastavené na výchozí hodnotu (MQPMO_NONE), bude k vloženým zprávám přidružena výchozí informace o kontextu.

Také způsob, jakým volání pracuje se synchronizovanými body, je určen platformou. Výchozí hodnota řízení synchronizačního bodu je v souboru z/OS ano; pro ostatní platformy, to není.

Kontext
Uvádí název popisovače fronty, ze kterého chcete kopírovat informace o kontextu (je-li požadováno v poli *Options*).

Úvod do kontextu zprávy viz “kontext zprávy” na stránce 45. Informace o použití struktury MQPMO k řízení informací o kontextu ve zprávě viz “Řízení informací o kontextu zprávy” na stránce 736.

ResolvedQName
Obsahuje název (po vyřešení libovolného názvu aliasu) fronty, která byla otevřena pro přijetí zprávy. Toto je výstupní pole.

Název ResolvedQMgr
Obsahuje název (po vyřešení libovolného názvu aliasu) správce front, který vlastní frontu v produktu *ResolvedQName*. Toto je výstupní pole.

MQPMO může také obsahovat pole požadovaná pro distribuční seznamy (viz “Distribuční seznamy” na stránce 739). Chcete-li použít tento prostředek, použije se verze 2 struktury MQPMO. To zahrnuje následující pole:

RecsPresent
Toto pole obsahuje počet front v distribučním seznamu, tj. počet záznamů MQPMR (Put Message Records) a odpovídajících záznamů odezvy (MQRR).

Hodnota, kterou zadáte, může být stejná jako počet záznamů objektů poskytnutých v MQOPEN. Pokud je však hodnota menší než počet záznamů objektů zadaný ve volání MQOPEN nebo pokud nezadáte žádné záznamy vložení zprávy, budou hodnoty nedefinovaných front převzaty z výchozích hodnot poskytnutých deskriptorem zprávy. Je-li hodnota větší než počet poskytnutých záznamů objektů, budou nadbytečné záznamy vložení zpráv ignorovány.

Doporučuje se provést jednu z následujících akcí:

- Chcete-li přijmout sestavu nebo odpověď z každého místa určení, zadejte stejnou hodnotu, jako je uvedena ve struktuře MQOR, a použijte MQPMR obsahující pole *MsgId* . Buď inicializujte tato pole *MsgId* na nuly, nebo uveďte MQPMO_NEW_MSG_ID.

  Po vložení zprávy do fronty budou v MQPMR k dispozici hodnoty *MsgId* vytvořené správcem front. Pomocí těchto hodnot můžete určit, které místo určení je přidruženo ke každé sestavě nebo odpovědi.
- Pokud nechcete přijímat sestavy nebo odpovědi, vyberte jednu z následujících možností:
  1. Chcete-li identifikovat místa určení, která okamžitě selžou, můžete i nadále zadat stejnou hodnotu do pole *RecsPresent* , jako je uvedeno ve struktuře MQOR, a poskytnout MQRRs pro identifikaci těchto míst určení. Neuvádějte žádné MQPMR.
  2. Pokud nechcete identifikovat nezdařené cíle, zadejte do pole *RecsPresent* nulu a neposkytujte MQPMR ani MQRRs.

Poznámka: Používáte-li produkt MQPUT1, musí být počet ukazatelů záznamu odezvy a posunutí záznamu odezvy nula.

Úplný popis záznamů vložení zprávy (MQPMR) a záznamů odezvy (MQRR) viz MQPMR a MQRR.

PutMsgRecFields
Označuje, která pole jsou přítomna v každém záznamu MQPMR (Put Message Record). Seznam těchto polí viz “Použití struktury MQPMR” na stránce 743.

PutMsgRecOffset* a *PutMsgRecPtr
Ukazatele (obvykle v jazyce C) a posuny (obvykle v jazyku COBOL) se používají k adresování záznamů vložení zprávy (přehled struktury MQPMR naleznete v části “Použití struktury MQPMR” na stránce 743).

Pomocí pole *PutMsgRecPtr* určete ukazatel na první záznam vložení zprávy nebo pomocí pole *PutMsgRecOffset* určete posun prvního záznamu vložení zprávy. Jedná se o posun od začátku MQPMO. V závislosti na poli *PutMsgRecFields* zadejte nenulovou hodnotu pro *PutMsgRecOffset* nebo *PutMsgRecPtr*.

ResponseRecOffset a ResponseRecPtr
K adresování záznamů odezvy se používají také ukazatele a posuny (další informace o záznamech odezvy naleznete v části “Použití struktury MQRR” na stránce 742).

Použijte pole *ResponseRecPtr* k uvedení ukazatele na první záznam odezvy, nebo pole *ResponseRecOffset* k uvedení odchylky prvního záznamu odezvy. Jedná se o posun od začátku struktury MQPMO. Zadejte nenulovou hodnotu pro *ResponseRecOffset* nebo *ResponseRecPtr*.

Poznámka: Pokud používáte MQPUT1 k vložení zpráv do distribučního seznamu, *ResponseRecPtr* musí mít hodnotu null nebo nula a *ResponseRecOffset* musí být nula.

Verze 3 struktury MQPMO dále obsahuje následující pole:

Popisovač OriginalMsg
Použití tohoto pole závisí na hodnotě pole *Akce* . Pokud vkládáte novou zprávu s přidruženými vlastnostmi zprávy, nastavte toto pole na popisovač zprávy, který jste předtím vytvořili, a nastavte vlastnosti. Pokud předáváte, odpovídáte nebo generujete sestavu jako odpověď na dříve načtenou zprávu, toto pole obsahuje popisovač zprávy této zprávy.

Popisovač NewMsg
Zadáte-li manipulátor *NewMsgManipulátor*, všechny vlastnosti přidružené k vlastnostem potlačení manipulátoru přidruženým k manipulátoru *OriginalMsgManipulátor*. Další informace viz Akce (MQLONG).

Akce
Toto pole použijte k určení typu prováděného vložení. Možné hodnoty a jejich význam jsou následující:

MQACTP_NEW
Toto je nová zpráva, která nesouvisí s žádnou jinou.

MQACTP_FORWARD
Tato zpráva byla načtena dříve a nyní je předávána.

MQACTP_REPLY
Tato zpráva je odpovědí na dříve načtenou zprávu.

MQACTP_REPORT
Tato zpráva je sestava vygenerovaná jako výsledek dříve načtené zprávy.

Další informace viz Akce (MQLONG).

PubLevel
Pokud je tato zpráva publikací, můžete toto pole nastavit tak, aby určilo, které odběry ji obdrží. Tuto publikaci obdrží pouze odběry s hodnotou *SubLevel* menší nebo rovnou této hodnotě. Výchozí hodnota je 9, což je nejvyšší úroveň, a znamená, že odběry s libovolnou *SubLevel* mohou obdržet toto publikování.

Data ve vaší zprávě

Zadejte adresu vyrovnávací paměti, která obsahuje data, do parametru **Buffer** volání MQPUT. Do dat ve zprávách můžete zahrnout cokoli. Množství dat ve zprávách však ovlivňuje výkon aplikace, která je zpracovává.

Maximální velikost dat je určena:

- Atribut **MaxMsgLength** správce front
- Atribut **MaxMsgLength** fronty, do které vkládáte zprávu
- Velikost libovolného záhlaví zprávy přidaného produktem IBM MQ (včetně záhlaví nedoručených zpráv, MQDLH a záhlaví distribučního seznamu, MQDH)

Atribut **MaxMsgLength** správce front obsahuje 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 pro objekt správce front. U velkých zpráv můžete tuto hodnotu změnit.

Atribut **MaxMsgLength** fronty určuje maximální velikost zprávy, kterou můžete vložit do fronty. Pokusíte-li se vložit zprávu o velikosti větší než je hodnota tohoto atributu, volání MQPUT se nezdaří. Vkládáte-li zprávu do vzdálené fronty, je maximální velikost zprávy, kterou lze úspěšně vložit, určena atributem **MaxMsgLength** vzdálené fronty, všemi prostředními přenosovými frontami, do kterých je zpráva vložena podél přenosové cesty k místu určení, a použitými kanály.

Pro operaci MQPUT musí být velikost zprávy menší nebo rovna atributu **MaxMsgLength** fronty i správce front. Hodnoty těchto atributů jsou nezávislé, ale doporučuje se nastavit *MaxMsgLength* fronty na hodnotu menší nebo rovnou hodnotě správce front.

Produkt IBM MQ přidává informace záhlaví ke zprávám 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 vlastnícího správce front.
- Pokud produkt IBM MQ nemůže doručit zprávu do vzdálené fronty, pokusí se vložit zprávu do fronty nedoručených zpráv (nedoručených zpráv). Přidá do zprávy strukturu MQDLH. Tato struktura zahrnuje název cílové fronty a důvod, proč byla zpráva vložena do fronty nedoručených zpráv.
- Chcete-li odeslat zprávu do více cílových front, produkt IBM MQ přidá do zprávy záhlaví MQDH. Popisuje data, která jsou přítomna ve zprávě, která patří do rozdělovníku, v přenosové frontě. Toto zvažte při výběru optimální hodnoty pro maximální délku zprávy.
- Je-li zpráva segmentem nebo zprávou ve skupině, může produkt IBM MQ přidat prostředí MQMDE.

Tyto struktury jsou popsány v části MQDH a MQMDE.

Pokud vaše zprávy mají maximální povolenou velikost pro tyto fronty, přidání těchto záhlaví znamená, že operace vložení se nezdaří, protože zprávy jsou nyní příliš velké. Chcete-li snížit možnost selhání operací vložení, postupujte takto:

- Zmenšete velikost zpráv, než je atribut **MaxMsgLength** přenosových front a front 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 atribut *MaxMsgLength* správce front, který vlastní frontu nedoručených zpráv.

Atributy pro správce front a konstanty řazení zpráv do front jsou popsány v části Atributy pro správce front.

z/OS Informace o způsobu zpracování nedoručených zpráv v prostředí distribuovaných front naleznete v tématu Nedoručené/nezpracované zprávy.

Vkládání zpráv: Použití popisovačů zpráv

Ve struktuře MQPMO jsou k dispozici dva manipulátory zpráv: *OriginalMsgHandle* a *NewMsgHandle*. Relace mezi těmito manipulátory zpráv je definována hodnotou pole *Akce* MQPMO.

Úplné podrobnosti viz Akce (MQLONG). Popisovač zprávy není nezbytně nutný pro vložení zprávy. Jeho účelem je přidružit vlastnosti ke zprávě, takže je vyžadováno pouze v případě, že používáte vlastnosti zprávy.

Vkládání zpráv do vzdálené fronty

Chcete-li vložit zprávu do vzdálené fronty (tj. do fronty vlastněné jiným správcem front než tou, ke které je připojena vaše aplikace), a nikoli do lokální fronty, je jediným dalším aspektem určení názvu fronty při jejím otevření.To je popsáno v tématu “Otevírání vzdálených front” na stránce 729. Způsob použití volání MQPUT nebo MQPUT1 pro lokální frontu není nijak změněn.

Další informace o použití vzdálených a přenosových front naleznete v tématu IBM MQ techniky distribuovaného řazení do front.

Nastavení vlastností zprávy

Pro každou vlastnost, kterou chcete nastavit, volejte MQSETMP. Když vložíte zprávu, nastavte popisovač zprávy a pole akce struktury MQPMO.

Chcete-li přidružit vlastnosti ke zprávě, musí mít zpráva popisovač zprávy. Vytvořte manipulátor zprávy pomocí volání funkce MQCRTMH. Pro každou vlastnost, kterou chcete nastavit, volejte MQSETMP s určením tohoto popisovače zprávy. Pro ilustraci použití MQSETMP je k dispozici ukázkový program amqsstma.c.

Pokud se jedná o novou zprávu, při jejím vložení do fronty pomocí příkazu MQPUT nebo MQPUT1nastavte pole obslužné rutiny OriginalMsgv MQPMO na hodnotu tohoto popisovače zprávy a pole Akce MQPMO na hodnotu MQACTP_NEW (toto je výchozí hodnota).

Jedná-li se o zprávu, kterou jste již načetli, a nyní ji přeposíláte nebo odpovídáte na ni nebo na ni odesíláte sestavu v reakci na ni, vložte původní popisovač zprávy do pole OriginalMsgpopisovače MQPMO a nový popisovač zprávy do pole NewMsgpopisovače. Podle potřeby nastavte pole Akce na hodnotu MQACTP_FORWARD, MQACTP_REPLY nebo MQACTP_REPORT.

Máte-li vlastnosti v záhlaví MQRFH2 ze zprávy, kterou jste načetli dříve, můžete je převést na vlastnosti popisovače zprávy pomocí volání MQBUFMH.

Pokud vkládáte zprávu do fronty ve správci front na úrovni starší než IBM WebSphere MQ 7.0, která nemůže zpracovat vlastnosti zprávy, můžete nastavit parametr PropertyControl v definici kanálu a určit způsob zpracování vlastností.

Řízení informací o kontextu zprávy

Při použití volání MQPUT nebo MQPUT1 k vložení zprávy do fronty můžete určit, že správce front má do deskriptoru zprávy přidat některé výchozí informace o kontextu. Aplikace, které mají odpovídající úroveň oprávnění, mohou přidat 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, zjistit informace o původci zprávy. Všechny informace o kontextu jsou uloženy v polích kontextu deskriptoru zprávy. Typ informací spadá do informací o identitě, původu a kontextu uživatele.

Chcete-li řídit informace o kontextu, použijte pole *Options* ve struktuře MQPMO.

Pokud nezadáte žádné volby pro informace o kontextu, správce front přepíše informace o kontextu, které již mohou být v deskriptoru zprávy, informacemi o identitě a kontextu, které vygeneroval pro vaši zprávu. Toto je stejné jako zadání volby MQPMO_DEFAULT_CONTEXT.Tuto výchozí informaci o kontextu můžete chtít při vytváření nové zprávy (například při zpracování uživatelského vstupu z dotazové obrazovky).

Nechcete-li ke zprávě přidružit žádné informace o kontextu, použijte volbu MQPMO_NO_CONTEXT. Při vkládání zprávy bez kontextu se veškeré kontroly oprávnění provedené produktem IBM MQ provádějí pomocí prázdného ID uživatele. Prázdnému ID uživatele nelze přiřadit explicitní oprávnění k prostředkům IBM MQ , ale je s ním zacházeno jako se členem speciální skupiny 'nikdo'. Další podrobnosti o speciální skupině nobodyviz Referenční informace o rozhraní instalovatelných služeb.

Nastavení kontextu můžete provést pomocí MQOPEN následované MQPUT pomocí volby MQOO_ a volby MQPMO_ uvedené v následujících sekcích. Nastavení kontextu můžete také provést pouze pomocí volby MQPUT1. V takovém případě stačí vybrat volbu MQPMO_ uvedenou v níže uvedených sekcích.

Následující sekce tohoto tématu vysvětlují použití kontextu identity, kontextu uživatele a celého kontextu.

- “Předávání kontextu identity” na stránce 736
- “Předávání uživatelského kontextu” na stránce 737
- “Předání celého kontextu” na stránce 737
- “Nastavení kontextu identity” na stránce 737
- “Nastavení uživatelského kontextu” na stránce 738
- “Nastavení celého kontextu” na stránce 738

Předávání kontextu identity

Obecně by programy měly předávat informace o kontextu identity ze zprávy do zprávy kolem aplikace, dokud data nedosáhnou svého konečného cíle.

Programy by měly změnit informace o původním kontextu pokaždé, když změní data. Avšak aplikace, které chtějí změnit nebo nastavit jakékoli informace o kontextu, musí mít odpovídající úroveň oprávnění. Správce front kontroluje toto oprávnění, když aplikace otevírají fronty; musí mít oprávnění k použití příslušných voleb kontextu pro volání MQOPEN.

Pokud aplikace obdrží zprávu, zpracuje data ze zprávy a poté změněná data vloží do jiné zprávy (pravděpodobně pro zpracování jinou aplikací), musí aplikace předat informace o kontextu identity z původní zprávy do nové zprávy. Správci front můžete povolit vytvoření informací o původním kontextu.

Chcete-li uložit informace o kontextu z původní zprávy, použijte volbu MQOO_SAVE_ALL_CONTEXT při otevření fronty pro získání zprávy. Jedná se o doplněk k dalším volbám, které používáte s voláním MQOPEN. Mějte však na paměti, že informace o kontextu nelze uložit, pouze pokud procházíte zprávu.

Při vytváření druhé zprávy:

- Otevřete frontu pomocí volby MQOO_PASS_IDENTITY_CONTEXT (kromě volby MQOO_OUTPUT).
- V poli *Context* struktury voleb vložení zprávy zadejte popisovač fronty, ze které jste uložili informace o kontextu.
- V poli *Options* struktury voleb put-message zadejte volbu MQPMO_PASS_IDENTITY_CONTEXT.

Předávání uživatelského kontextu

Nelze zvolit předání pouze uživatelského kontextu. Chcete-li při vkládání zprávy předat uživatelský kontext, zadejte hodnotu MQPMO_PASS_ALL_CONTEXT. Všechny vlastnosti v uživatelském kontextu jsou předávány stejným způsobem jako původní kontext.

Když dojde k MQPUT nebo MQPUT1 a předává se kontext, předají se všechny vlastnosti v uživatelském kontextu z načtené zprávy do vkládané zprávy. Všechny vlastnosti uživatelského kontextu, které vkládající aplikace změnila, jsou vloženy s jejich původními hodnotami. Všechny vlastnosti uživatelského kontextu, které vkládající aplikace odstranila, jsou obnoveny ve zprávě vložení. Všechny vlastnosti uživatelského kontextu, které vkládající aplikace přidala do zprávy, jsou zachovány.

Předání celého kontextu

Pokud aplikace obdrží zprávu a vloží data zprávy (beze změny) do jiné zprávy, aplikace musí předat všechny informace o kontextu (identita, původ a uživatel) z původní zprávy do nové zprávy. Příkladem aplikace, která to může provést, je modul pro přesouvání zpráv, který přesouvá zprávy z jedné fronty do jiné.

Postupujte stejně jako v případě předání kontextu identity s tím rozdílem, že použijete volbu MQOPEN MQOO_PASS_ALL_CONTEXT a volbu put-message MQPMO_PASS_ALL_CONTEXT.

Nastavení kontextu identity

Chcete-li nastavit informace o kontextu identity pro zprávu:

- Otevřete frontu pomocí volby MQOO_SET_IDENTITY_CONTEXT.
- Vložte zprávu do fronty a zadejte volbu MQPMO_SET_IDENTITY_CONTEXT. V deskriptoru zprávy zadejte požadované informace o kontextu identity.

Poznámka: Když nastavíte některá (ale ne všechna) 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 vydání volání. Chcete-li například použít MQOOO_SET_IDENTITY_CONTEXT nebo MQPMO_SET_IDENTITY_CONTEXT, musíte mít oprávnění `+setid` .

Nastavení uživatelského kontextu

Chcete-li nastavit vlastnost v uživatelském kontextu, nastavte pole Kontext deskriptoru vlastnosti zprávy (MQPD) při volání MQSETMP na hodnotu MQPD_USER_CONTEXT.

K nastavení vlastnosti v uživatelském kontextu nepotřebujete žádné speciální oprávnění. Uživatelský kontext nemá žádné volby kontextu MQOO_SET_ * nebo MQPMO_SET_ *.

Nastavení celého kontextu

Chcete-li pro zprávu nastavit informace o identitě i o původním kontextu, postupujte takto:

1. Otevřete frontu pomocí volby MQOO_SET_ALL_CONTEXT.
2. Vložte zprávu do fronty a zadejte volbu MQPMO_SET_ALL_CONTEXT. V deskriptoru zprávy zadejte požadované informace o identitě a původu kontextu.

Pro každý typ nastavení kontextu je zapotřebí odpovídající oprávnění.

Související pojmy

“kontext zprávy” na stránce 45
Kontext zprávy umožňuje aplikaci, která načte zprávu, zjistit informace o původci zprávy.

Související odkazy

“Volby MQOPEN týkající se kontextu zprávy” na stránce 727
Chcete-li mít možnost přidružit informace o kontextu ke zprávě při jejím vložení do fronty, musíte při otevření fronty použít jednu z voleb kontextu zprávy.

Vložení jedné zprávy do fronty pomocí volání MQPUT1

Volání MQPUT1 použijte v případě, že chcete zavřít frontu ihned po vložení jediné zprávy. Například serverová aplikace pravděpodobně použije volání MQPUT1 , když odesílá odpověď do každé z různých front.

MQPUT1 je funkčně ekvivalentní volání MQOPEN následované MQPUT následovaným MQCLOSE. Jediný rozdíl v syntaxi volání MQPUT a MQPUT1 spočívá v tom, že pro MQPUT určíte popisovač objektu, zatímco pro MQPUT1 určíte strukturu deskriptoru objektu (MQOD), jak je definováno v MQOPEN (viz “Identifikace objektů (struktura MQOD)” na stránce 722). Je to proto, že musíte poskytnout informace volání MQPUT1 o frontě, kterou má otevřít, zatímco když voláte MQPUT, musí být fronta již otevřená.

Jako vstup pro volání MQPUT1 musíte zadat:

- Manipulátor připojení.
- Popis objektu, který chcete otevřít. Toto je ve formě struktury deskriptoru objektu (MQOD).
- Popis zprávy, kterou chcete vložit do fronty. Jedná se o strukturu deskriptoru zpráv (MQMD).
- Řídicí informace ve formě struktury voleb vložení zprávy (MQPMO).
- Délka dat obsažených ve zprávě (MQLONG).
- Adresa dat zprávy.

Výstup příkazu MQPUT1 je:

- Kód dokončení
- Kód příčiny

Pokud je volání úspěšně dokončeno, vrátí také strukturu voleb a strukturu deskriptoru zpráv. Volání upraví strukturu voleb tak, aby zobrazovala název fronty a správce front, do kterého byla zpráva odeslána. Pokud požadujete, aby správce front vygeneroval jedinečnou hodnotu pro identifikátor zprávy, kterou vkládáte (zadáním binární nuly do pole *MsgId* struktury MQMD), volání před vrácením této struktury vloží hodnotu do pole *MsgId* .

Poznámka: Produkt MQPUT1 nelze použít s názvem modelové fronty. Po otevření modelové fronty však můžete pro dynamickou frontu zadat příkaz MQPUT1 .

Šest vstupních parametrů pro MQPUT1 je:

Hconn
Jedná se o manipulátor připojení. Pro aplikace 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 manipulátor připojení vrácený 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 tuto frontu vlastní.

Pole *DynamicQName* je pro volání MQPUT1 ignorováno, protože nemůže používat modelové fronty.

Chcete-li nominovat alternativní identifikátor uživatele, který se má použít k testování oprávnění k otevření fronty, použijte pole *AlternateUserId* .

MsgDesc
Jedná se o strukturu deskriptoru zpráv (MQMD). Stejně jako u volání MQPUT definujte pomocí této struktury zprávu, kterou vkládáte do fronty.

PutMsgOpts
Jedná se o strukturu voleb vložení zprávy (MQPMO). Použijte jej jako pro volání MQPUT (viz “Určení voleb pomocí struktury MQPMO” na stránce 732).

Je-li pole *Options* nastaveno na nulu, správce front použije při provádění testů oprávnění pro přístup k frontě vaše vlastní ID uživatele. Správce front také ignoruje jakýkoli alternativní identifikátor uživatele zadaný v poli *AlternateUserId* struktury MQOD.

BufferLength
Toto je délka vaší zprávy.

Buffer
Toto je oblast vyrovnávací paměti, která obsahuje text zprávy.

Používáte-li klastry, produkt MQPUT1 funguje, jako by byla v platnosti volba MQOO_BIND_NOT_FIXED. Aplikace musí použít vyřešená pole ve struktuře MQPMO, a nikoli strukturu MQOD, aby určily, kam byla zpráva odeslána. Další informace naleznete v tématu Konfigurace klastru správců front .

V tabulce MQPUT1je uveden popis volání MQPUT1 .

Distribuční seznamy

Není podporováno na systému IBM MQ for z/OS. Distribuční seznamy umožňují vložit zprávu do více míst určení v jednom volání MQPUT nebo MQPUT1 .Jedno volání MQOPEN může otevřít více front a jedno volání MQPUT pak může do každé z těchto front vložit zprávu. Některé obecné informace ze struktur MQI použitých pro tento proces mohou být nahrazeny specifickými informacemi týkajícími se jednotlivých míst určení uvedených v distribučním seznamu.

Upozornění: Distribuční seznamy nepodporují použití alias fronta, které odkazují na objekty tématu. Pokud fronta aliasů odkazuje na objekt tématu v distribučním seznamu, IBM MQ vrátí MQRC_ALIAS_BASE_Q_TYPE_ERROR.

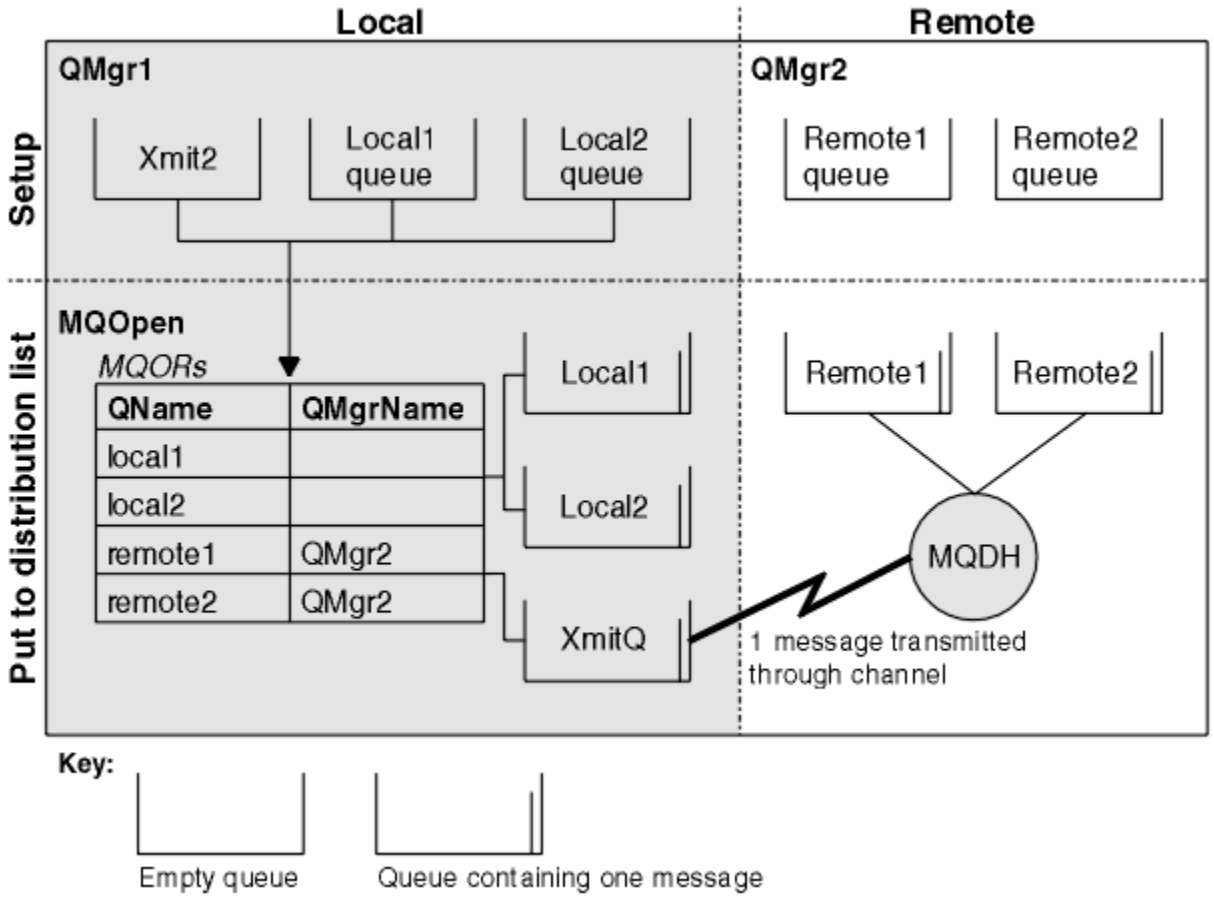
Když je vydáno volání MQOPEN, generické informace jsou převzaty z deskriptoru objektu (MQOD). Zadáte-li do pole *Version* hodnotu MQOD_VERSION_2 a do pole *RecsPresent* hodnotu větší než nula, lze *Hobj* definovat jako popisovač seznamu (jedné nebo více front) a nikoli jako popisovač fronty. V tomto případě jsou specifické informace poskytovány prostřednictvím záznamů objektů (MQORs), které poskytují podrobnosti o cíli (tj. *ObjectName* a *ObjectQMgrName*).

Manipulátor objektu (*Hobj*) je předán volání MQPUT, což vám umožňuje umístit jej do seznamu, nikoli do jedné fronty.

Při vkládání zprávy do front (MQPUT) jsou ze struktury MQPMO (Put Message Option structure) a deskriptoru MQMD (Message Descriptor) převzaty generické informace. Specifické informace jsou uvedeny ve formě záznamů MQPMR (Put Message Records).

Záznamy odpovědí (MQRR) mohou obdržet kód dokončení a kód příčiny specifický pro každou cílovou frontu.

Obrázek 56 na stránce 740 ukazuje, jak distribuční seznamy fungují.



Obrázek 56. Jak fungují distribuční seznamy

Otevírání distribučních seznamů
Pomocí volání MQOPEN otevřete distribuční seznam a pomocí voleb volání určete, co chcete se seznamem provést.

Jako vstup pro MQOPEN musíte dodat:

- Manipulátor připojení (popis viz “Vkládání zpráv do fronty” na stránce 730).
- Generické informace ve struktuře deskriptoru objektu (MQOD)
- Název každé fronty, kterou chcete otevřít, pomocí struktury záznamu objektu (MQOR).

Výstup z MQOPEN je:

- Popisovač objektu, který představuje váš přístup k rozdělovníku.
- Generický kód dokončení
- Generický kód příčiny
- Záznamy odpovědí (volitelné) obsahující kód dokončení a příčinu pro každé místo určení.

Použití struktury MQOD

Pomocí struktury MQOD identifikujte fronty, které chcete otevřít.

Chcete-li definovat distribuční seznam, musíte do pole *Version* zadat hodnotu MQOD_VERSION_2 , do pole *RecsPresent* hodnotu větší než nula a do pole *ObjectType* hodnotu MQOT_Q. Popis všech polí struktury MQOD viz MQOD .

Použití struktury MQOR

Zadejte strukturu MQOR pro každé místo určení.

Struktura obsahuje názvy cílových front a správců front. Pole *ObjectName* a *ObjectQMgrName* v produktu MQOD se nepoužívají pro distribuční seznamy. Musí existovat jeden nebo více záznamů objektů. Pokud je parametr *ObjectQMgrName* ponechán prázdný, použije se lokální správce front. Další informace o těchto polích viz ObjectName a ObjectQMgrName .

Cílové fronty můžete zadat dvěma způsoby:

- Pomocí pole posunutí *ObjectRecOffset*.

  V tomto případě musí aplikace deklarovat svou vlastní strukturu obsahující strukturu MQOD, následovanou polem záznamů MQOR (s tolika prvky pole, kolik je potřeba), a nastavit *ObjectRecOffset* na offset prvního prvku v poli od začátku MQOD. Ujistěte se, že je toto posunutí správné.

  Doporučuje se používat vestavěné prostředky poskytované 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 použijte konstantu MQOD_CURRENT_LENGTH, pokud programovací jazyk nepodporuje nezbytné vestavěné prostředky ve všech příslušných prostředích. Tuto techniku ilustruje následující kód:

```
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 přeskakovací bajty mezi MQOD a pole MQOR, musí být tyto bajty přidány k hodnotě uložené v souboru *ObjectRecOffset*.

  Použití *ObjectRecOffset* se doporučuje pro programovací jazyky, které nepodporují datový typ ukazatele, nebo které implementují datový typ ukazatele způsobem, který není přenositelný do různých prostředí (například programovací jazyk COBOL).

- Pomocí pole ukazatele *ObjectRecPtr*.

  V tomto případě může aplikace deklarovat pole struktur 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 přenositelný do různých prostředí (například programovací jazyk C).

Bez ohledu na techniku, kterou zvolíte, musíte použít jeden z *ObjectRecOffset* a *ObjectRecPtr* ; volání selže s kódem příčiny MQRC_OBJECT_RECORDS_ERROR, jsou-li obě nulové, nebo jsou-li nenulové.

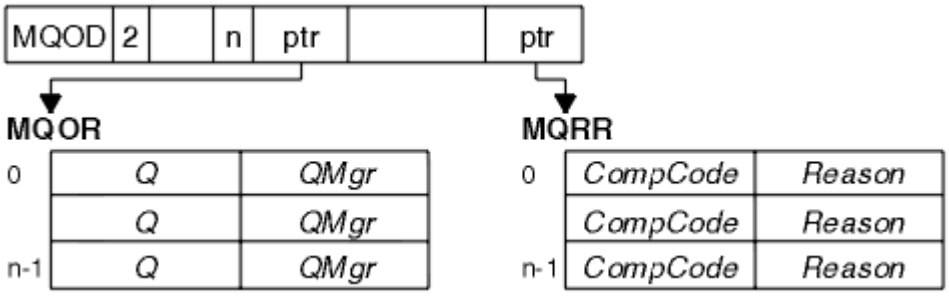
Použití struktury MQRR

Tyto struktury jsou specifické pro cíl; každý záznam odezvy obsahuje pole *CompCode* a *Reason* pro každou frontu distribučního seznamu. Musíte použít tuto strukturu, abyste mohli rozlišit, kde jsou problémy.

Pokud například obdržíte kód příčiny MQRC_MULTIPLE_REASON a váš distribuční seznam obsahuje pět cílových front, nebudete vědět, na které fronty se problémy vztahují, pokud tuto strukturu nepoužijete. Máte-li však kód dokončení a kód příčiny pro každé místo určení, můžete chyby vyhledat snadněji.

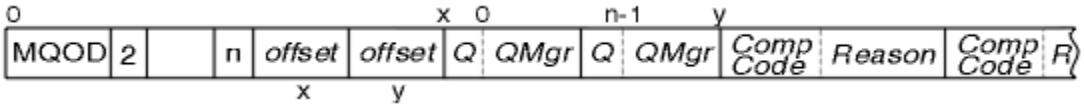
Další informace o struktuře MQRR viz MQRR .

Obrázek 57 na stránce 742 ukazuje, jak můžete otevřít distribuční seznam v C.



Obrázek 57. Otevření distribučního seznamu v C

Obrázek 58 na stránce 742 ukazuje, jak můžete otevřít rozdělovník v COBOLu.



Obrázek 58. Otevření rozdělovníku v COBOLu

Použití voleb MQOPEN

Při otevírání rozdělovníku můžete zadat následující volby:

- MQOO_OUTPUT
- MQOO_FAIL_IF_QUIESCING (volitelné)
- MQOO_ALTERNATE_USER_AUTHORITY (volitelné)
- MQOO_ * _CONTEXT (volitelné)

Popis těchto voleb viz “Otevírání a zavírání objektů” na stránce 720 .

Vložení zpráv do distribučního seznamu
Chcete-li vložit zprávy do distribučního seznamu, můžete použít příkaz MQPUT nebo MQPUT1.

Jako vstup musíte dodat:

- Manipulátor připojení (popis viz “Vkládání zpráv do fronty” na stránce 730).
- Popisovač objektu. Pokud je distribuční seznam otevřen pomocí MQOPEN, produkt *Hobj* umožňuje pouze jeho vložení do seznamu.
- Struktura deskriptoru zpráv (MQMD). Popis této struktury viz MQMD .
- Řídicí informace ve formě struktury volby vložení zprávy (MQPMO). Informace o vyplnění polí struktury MQPMO viz “Určení voleb pomocí struktury MQPMO” na stránce 732 .
- Řídicí informace ve formě záznamů MQPMR (Put Message Records).
- 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 poskytuje informace specifické pro cíl pro některá pole, která můžete chtít identifikovat jinak než ta, která jsou již identifikována v deskriptoru MQMD.

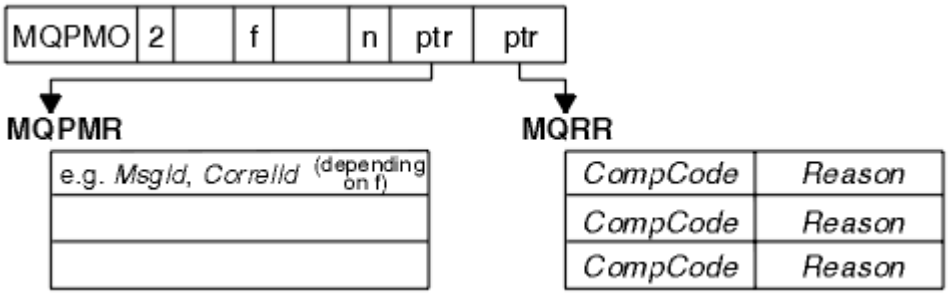
Popis těchto polí viz MQPMR.

Obsah každého záznamu závisí na informacích uvedených v poli *PutMsgRecFields* MQPMO. Například v ukázkovém programu AMQSPTL0.C (popis viz "Ukázkový program distribučního seznamu" na stránce 1053) zobrazující použití distribučních seznamů, ukázka se rozhodne poskytnout hodnoty 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;
```

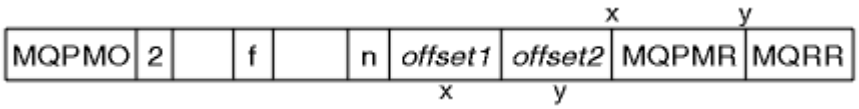
To znamená, že *MsgId* a *CorrelId* jsou poskytnuty pro každý cíl distribučního seznamu. Záznamy vložení zprávy jsou poskytovány jako pole.

Obrázek 59 na stránce 743 ukazuje, jak můžete vložit zprávu do distribučního seznamu v C.



Obrázek 59. Vložení zprávy do distribučního seznamu v jazyce C

Obrázek 60 na stránce 743 ukazuje, jak můžete vložit zprávu do rozdělovníku v COBOLu.



Obrázek 60. Vložení zprávy do distribučního seznamu v COBOLu

Použití MQPUT1

Používáte-li produkt MQPUT1, zvažte následující body:

1. Hodnoty polí *ResponseRecOffset* a *ResponseRecPtr* musí být null nebo nula.
2. Záznamy odezvy musí být v případě potřeby adresovány z produktu MQOD.

Některé případy, kdy volání put selhávají

Pokud jsou určité atributy fronty změněny pomocí volby FORCE v příkazu během intervalu mezi zadáním volání MQOPEN a volání MQPUT, volání MQPUT selže a vrátí kód příčiny MQRC_OBJECT_CHANGED.

Správce front označí popisovač objektu jako neplatný. K tomu dochází také v případě, že jsou změny provedeny během zpracování volání MQPUT1 nebo pokud se změny vztahují na libovolnou frontu, pro

kterou je název fronty přeložen. Atributy, které tímto způsobem ovlivňují manipulátor, jsou uvedeny v popisu volání MQOPEN v části MQOPEN. Pokud volání vrátí kód příčiny MQRC_OBJECT_CHANGED, zavřete frontu, znovu ji otevřete a pokuste se znovu vložit zprávu.

Pokud jsou operace vložení zablokovány pro frontu, do které se pokoušíte vložit zprávy (nebo libovolnou frontu, do které se název fronty interpretuje), volání MQPUT nebo MQPUT1 selže a vrátí kód příčiny MQRC_PUT_INHIBITED. Můžete být schopni úspěšně vložit zprávu, pokud se pokusíte o volání později, pokud je návrh aplikace takový, že ostatní programy pravidelně mění atributy front.

Dále platí, že pokud je fronta, do které se pokoušíte vložit zprávu, plná, volání MQPUT nebo MQPUT1 selže a vrátí hodnotu MQRC_Q_FULL.

Pokud byla odstraněna dynamická fronta (dočasná nebo trvalá), volání MQPUT s použitím dříve získaného popisovače objektu se nezdaří a vrátí kód příčiny MQRC_Q_DELETED. V této situaci je dobrým zvykem zavřít popisovač objektu, protože pro vás již není k ničemu.

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 vložení zpráv do více míst určení, budou záznamy odezvy obsahovat specifické položky *CompCode* a *Reason* pro každé místo určení. Pokud obdržíte kód dokončení MQCC_FAILED, nebude do žádné cílové fronty úspěšně vložena žádná zpráva. Je-li kód dokončení MQCC_WARNING, zpráva je úspěšně vložena do jedné nebo více cílových front. Pokud obdržíte návratový kód MQRC_MULTIPLE_REASON, nejsou všechny kódy příčiny stejné pro každé místo určení. Proto se doporučuje použít strukturu MQRR, abyste mohli určit, která fronta nebo fronty způsobily chybu a příčiny pro každou z nich.

Získávání zpráv z fronty

Pomocí těchto informací získáte informace o získávání zpráv z fronty.

Zprávy z fronty můžete získat dvěma způsoby:

1. Můžete odebrat zprávu z fronty, aby ji ostatní programy již neviděli.
2. Můžete zkopírovat zprávu a ponechat původní zprávu ve frontě. Toto se nazývá *procházení*. Zprávu můžete odebrat, jakmile ji procházíte.

V obou případech použijete 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 části “Připojení ke správci front a odpojení od něj” na stránce 713 a “Otevírání a zavírání objektů” na stránce 720.

Po otevření fronty můžete opakovaně používat volání MQGET k procházení nebo odebírání zpráv ve stejné frontě. Po dokončení načítání všech požadovaných zpráv z fronty volejte příkaz MQCLOSE.

Další informace o získávání zpráv z fronty naleznete v následujících odkazech:

- “Získávání zpráv z fronty pomocí volání MQGET” na stránce 745
- “Pořadí, ve kterém jsou zprávy načítány z fronty” na stránce 749
- “Získání konkrétní zprávy” na stránce 760
- “Zlepšení výkonu dočasných zpráv” na stránce 761
- z/OS “Typ indexu” na stránce 765
- “Zpracování zpráv delších než 4 MB” na stránce 766
- “Čekání na zprávy” na stránce 771
- z/OS “signalizace” na stránce 772
- “Přeskočení vrácení” na stránce 773
- “Převod dat aplikace” na stránce 775
- “Procházení zpráv ve frontě” na stránce 776

- “Některé případy, kdy volání MQGET selže” na stránce 781

Související pojmy

“Přehled rozhraní fronty zpráv” na stránce 700
Informace o komponentách rozhraní MQI (Message Queue Interface).

“Připojení ke správci front a odpojení od něj” na stránce 713
Chcete-li používat programovací služby IBM MQ , musí mít program připojení ke správci front. Pomocí těchto informací se dozvíte, jak se připojit ke správci front a jak se od něj odpojit.

“Otevírání a zavírání objektů” na stránce 720
Tyto informace poskytují náhled na otevírání a zavírání objektů IBM MQ .

“Vkládání zpráv do fronty” na stránce 730
Pomocí těchto informací se dozvíte, jak vkládat zprávy do fronty.

“Zjišťování a nastavení atributů objektu” na stránce 822
Atributy jsou vlastnosti, které definují charakteristiky objektu IBM MQ .

“Potvrzení a zálohování jednotek práce” na stránce 825
Tyto informace popisují, jak potvrdit a vrátit zpět všechny obnovitelné operace get a put, které se vyskytly v transakci.

“Spuštění aplikací IBM MQ pomocí spouštěčů” na stránce 836
Získejte informace o spouštěčích a o tom, jak spouštět aplikace IBM MQ pomocí spouštěčů.

“Práce s rozhraním MQI a klastry” na stránce 854
Existují speciální volby pro volání a návratové kódy, které se týkají klastrování.

“Používání a psaní aplikací na systému IBM MQ for z/OS” na stránce 858
Aplikace IBM MQ for z/OS lze vytvořit z programů, které jsou spuštěny v mnoha různých prostředích. To znamená, že mohou využívat zařízení, která jsou k dispozici ve více než jednom prostředí.

“IMS a IMS přemostění aplikací na IBM MQ for z/OS” na stránce 66
Tyto informace vám pomohou psát aplikace IMS pomocí IBM MQ.

Získávání zpráv z fronty pomocí volání MQGET

Volání MQGET získá zprávu z otevřené lokální fronty. Nemůže získat zprávu z fronty na jiném systému.

Jako vstup pro volání MQGET musíte zadat:

- Manipulátor připojení.
- Popisovač fronty.
- Popis zprávy, kterou chcete získat z fronty. Jedná se o strukturu deskriptoru zpráv (MQMD).
- Řídicí informace ve formě struktury MQGMO (Get Message Options).
- Velikost vyrovnávací paměti, kterou jste přiřadili k zadržení zprávy (MQLONG).
- Adresa úložiště, do kterého má být zpráva vložena.

Výstup příkazu MQGET je:

- Kód příčiny
- Kód dokončení
- Zpráva v oblasti vyrovnávací paměti, kterou jste zadali, pokud bylo volání úspěšně dokončeno.
- Struktura voleb upravená tak, aby zobrazovala název fronty, ze které byla zpráva načtena.
- Struktura deskriptoru zprávy s obsahem polí upravených tak, aby popisovala načtenou zprávu.
- Délka zprávy (MQLONG)

V části MQGETje uveden popis volání MQGET.

Následující sekce popisují informace, které musíte zadat jako vstup pro volání MQGET.

- “Určení manipulátorů připojení” na stránce 746
- “Popis zpráv pomocí struktury MQMD a volání MQGET” na stránce 746

- "Určení voleb MQGET pomocí struktury MQGMO" na stránce 746
- "Určení velikosti oblasti vyrovnávací paměti" na stránce 748

Určení manipulátorů připojení

z/OS Pro aplikace systému CICS on z/OS můžete určit konstantu MQHC_DEF_HCONN (která má hodnotu nula) nebo použít manipulátor připojení vrácený voláním MQCONN nebo MQCONNX. Pro ostatní aplikace vždy používejte manipulátor připojení vrácený voláním MQCONN nebo MQCONNX.

Použijte popisovač fronty (*Hobj*), který je vrácen při volání MQOPEN.

Popis 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. K dispozici je úvod do vlastností zprávy, které modul MQMD popisuje v souboru "Zprávy produktu IBM MQ" na stránce 18, a popis samotné struktury v modulu MQMD.

Pokud víte, kterou zprávu chcete získat z fronty, viz "Získání konkrétní zprávy" na stránce 760.

Pokud neurčíte konkrétní zprávu, příkaz MQGET načte *první* zprávu ve frontě. "Pořadí, ve kterém jsou zprávy načítány z fronty" na stránce 749 popisuje, jak 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 pro procházení zpráv ve frontě), musíte po každém volání nastavit pole *MsgId* a *CorrelId* této struktury na hodnotu null. Toto vymaže tato pole identifikátorů zprávy, která byla načtena.

Chcete-li však své zprávy seskupit, musí být hodnota *GroupId* stejná pro zprávy ve stejné skupině, aby volání vyhledalo zprávu se stejnými identifikátory jako předchozí zpráva, aby bylo možné vytvořit celou skupinu.

Určení voleb MQGET pomocí struktury MQGMO

Struktura MQGMO je vstupní/výstupní proměnná pro předávání voleb do volání MQGET. Následující sekce vám pomohou dokončit některá pole této struktury.

V části MQGMOje uveden popis struktury MQGMO.

StrucId
StrucId je 4znakové pole používané k identifikaci struktury jako struktury voleb get-message. Vždy zadejte MQGMO_STRUC_ID.

Version
*Version*popisuje číslo verze struktury. MQGMO_VERSION_1 je výchozí. Chcete-li použít pole verze 2 nebo načíst zprávy v logickém pořadí, zadejte MQGMO_VERSION_2. Chcete-li použít pole verze 3 nebo načíst zprávy v logickém pořadí, zadejte hodnotu MQGMO_VERSION_3. MQGMO_CURRENT_VERSION nastaví aplikaci tak, aby používala nejnovější úroveň.

Options
V rámci kódu můžete vybrat volby v libovolném pořadí; každá volba je v poli *Options* reprezentována bitem.

Ovládací prvky pole *Options* :

- Zda volání MQGET čeká na doručení zprávy do fronty před jejím dokončením (viz "Čekání na zprávy" na stránce 771).
- Zda je operace získání zahrnuta do pracovní jednotky.
- Zda je dočasná 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 přeskakující vrácení (viz "Přeskočení vrácení" na stránce 773).

- Zda je zpráva odebrána z fronty, nebo zda je pouze procházena
- Zda vybrat zprávu pomocí kurzoru procházení nebo jiných kritérií výběru
- Zda je volání úspěšné, i když je zpráva delší než vaše vyrovnávací paměť.
- z/OS V systému IBM MQ for z/OS, zda povolit dokončení volání. Tato volba také nastaví signál, který označuje, že chcete být upozorněni, když přijde zpráva
- Zda se volání nezdaří, pokud je správce front ve stavu uvedení do klidového stavu
- z/OS V systému IBM MQ for z/OS, zda volání selže, pokud je připojení v klidovém stavu
- Zda je vyžadován převod dat zpráv aplikace (viz “Převod dat aplikace” na stránce 775).
- Pořadí, ve kterém jsou zprávy a segmenty načítány z fronty z/OS (kromě IBM MQ for z/OS)
- Zda jsou úplné, pouze logické zprávy, které lze načíst z/OS (s výjimkou IBM MQ for z/OS)
- Zda lze zprávy ve skupině načíst pouze v případě, že jsou k dispozici *všechny* zprávy ve skupině
- Zda lze segmenty v logické zprávě načíst pouze v případě, že jsou k dispozici *všechny* segmenty v logické zprávě z/OS (s výjimkou IBM MQ for z/OS).

Ponecháte-li pole *Options* nastavené na výchozí hodnotu (MQGMO_NO_WAIT), bude volání MQGET pracovat tímto způsobem:

- Pokud vašemu kritériu výběru ve frontě neodpovídá žádná zpráva, volání nečeká na doručení zprávy, ale dokončí se okamžitě. z/OS Také v IBM MQ for z/OSvolání nenastavuje signál požadující oznámení, když taková zpráva dorazí.
- Způsob, jakým volání pracuje se synchronizačním bodem, je určen platformou:

Platforma	**Pod řízením synchronizačního bodu**
IBM i	Ne
Systémy AIX 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 přeskakující vrácení.
- Vybraná zpráva je odebrána z fronty (není procházena).
- Není vyžadován žádný převod dat zprávy aplikace.
- Volání selže, pokud je zpráva delší než vaše vyrovnávací paměť.

WaitInterval

Pole *WaitInterval* určuje maximální dobu (v milisekundách), po kterou volání MQGET čeká na doručení zprávy do fronty při použití volby MQGMO_WAIT. Pokud v době uvedené v parametru *WaitInterval*nedorazí žádná zpráva, volání se dokončí a vrátí kód příčiny, který by ukazoval, že ve frontě nebyla žádná zpráva, která by odpovídala kritériím výběru.

z/OS Pokud v systému IBM MQ for z/OSpoužijete volbu MQGMO_SET_SIGNAL, pole *WaitInterval* určuje čas, po který je signál nastaven.

Další informace o těchto volbách viz “Čekání na zprávy” na stránce 771 z/OS a “signalizace” na stránce 772 .

z/OS ***Signal1***

Volba Signal1 je podporována pouze v systému IBM MQ for z/OS.

Pokud pomocí volby MQGMO_SET_SIGNAL požadujete, aby byla aplikace upozorněna při doručení vhodné zprávy, zadejte typ signálu do pole *Signal1* . V systému IBM MQ na všech ostatních platformách je pole *Signal1* vyhrazeno a jeho hodnota není významná.

z/OS Další informace viz téma “signalizace” na stránce 772.

Signal2
Pole *Signal2* je vyhrazeno na všech platformách a jeho hodnota není významná.

z/OS Další informace viz “signalizace” na stránce 772.

ResolvedQName
ResolvedQName je výstupní pole, ve kterém správce front vrací název fronty (po vyřešení libovolného aliasu), ze které byla zpráva načtena.

MatchOptions
Volba *MatchOptions* řídí kritéria výběru pro příkaz MQGET.

GroupStatus
GroupStatus označuje, zda je zpráva, kterou jste načetli, ve skupině.

SegmentStatus
SegmentStatus označuje, zda je načtená položka segmentem logické zprávy.

Segmentation
Segmentation označuje, zda je pro načtenou zprávu povolena segmentace.

MsgToken

MsgToken Jedinečně identifikuje zprávu.

ReturnedLength
ReturnedLength je výstupní pole, ve kterém správce front vrací délku vrácených dat zprávy (v bajtech).

MsgHandle
Popisovač zprávy, který má být naplněn vlastnostmi zprávy načítané z fronty. Manipulátor byl dříve vytvořen voláním MQCRTMH. Všechny vlastnosti, které jsou již k popisovači přidruženy, jsou před načtením zprávy vymazány.

Určení velikosti oblasti vyrovnávací paměti

V parametru **BufferLength** volání MQGET zadejte velikost oblasti vyrovnávací paměti pro uchování dat zprávy, která načtete. Můžete se rozhodnout, jak velký by to mělo být třemi způsoby:

1. Možná již víte, jakou délku zpráv od tohoto programu očekáváte. Pokud ano, zadejte vyrovnávací paměť této velikosti.

   Můžete však použít volbu MQGMO_ACCEPT_TRUNCATED_MSG ve struktuře MQGMO, chcete-li, aby bylo volání MQGET dokončeno, i když je zpráva pro vyrovnávací paměť příliš velká. V tomto případě:

   - Vyrovnávací paměť je naplněna tak velkou část zprávy, jak může zadržet
   - Volání vrátí kód dokončení varování.
   - Zpráva je odebrána z fronty (vyřazení zbytku zprávy) nebo je kurzor procházení rozšířen (pokud procházíte frontu).
   - Skutečná délka zprávy je vrácena v *DataLength*

   Bez této volby se volání stále dokončí s varováním, ale neodebere zprávu z fronty (nebo přesune kurzor procházení).
2. Odhadněte velikost vyrovnávací paměti (nebo dokonce uveďte velikost nula bajtů) a *nepoužívejte* volbu MQGMO_ACCEPT_TRUNCATED_MSG. Pokud volání MQGET selže (například proto, že vyrovnávací paměť je příliš malá), vrátí se délka zprávy v parametru **DataLength** volání. (Vyrovnávací paměť je stále naplněna co největší část zprávy, ale zpracování volání není dokončeno.) Uložte soubor *MsgId*

této zprávy a poté zopakujte volání MQGET s určením oblasti vyrovnávací paměti správné velikosti a hodnoty *MsgId* , které jste zaznamenali při prvním volání.

Pokud váš program obsluhuje frontu, která je obsluhována také jinými programy, jeden z těchto programů může odebrat zprávu, kterou chcete, než váš program může vydat další volání MQGET. Váš program může ztrácet čas hledáním zprávy, která již neexistuje. Chcete-li se tomu vyhnout, nejprve procházejte frontu, dokud nenaleznete požadovanou zprávu, přičemž uveďte hodnotu *BufferLength* nula a použijte volbu MQGMO_ACCEPT_TRUNCATED_MSG. Tato volba umístí kurzor procházení pod požadovanou zprávu. Poté můžete zprávu načíst opětovným voláním příkazu MQGET a zadáním volby MQGMO_MSG_UNDER_CURSOR. Pokud jiný program odebere zprávu mezi voláním procházení a odebráním, druhý příkaz MQGET okamžitě selže (bez prohledávání celé fronty), protože pod 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 očekáváte, můžete se dotázat na atribut **MaxMsgLength** (pomocí volání MQINQ) a poté zadat vyrovnávací paměť této velikosti.

   Pokuste se velikost vyrovnávací paměti co nejvíce přiblížit skutečné velikosti zprávy, abyste se vyhnuli snížení výkonu.

   Další informace o atributu **MaxMsgLength** viz “Zvýšení maximální délky zprávy” na stránce 766.

Pořadí, ve kterém jsou zprávy načítány z fronty

Můžete řídit pořadí, ve kterém načítáte zprávy z fronty. Tato část se zabývá možnostmi.

Priorita

Program může přiřadit prioritu zprávě, když ji umístí do fronty (viz “Priority zpráv” na stránce 25). Zprávy se stejnou prioritou jsou uloženy ve frontě v pořadí přijetí, nikoli v pořadí, ve kterém jsou potvrzeny.

Správce front udržuje fronty buď ve striktním pořadí FIFO (první dovnitř, první ven), nebo v pořadí FIFO v rámci priority. To závisí na nastavení atributu **MsgDeliverySequence** fronty. Když zpráva dorazí do fronty, je vložena bezprostředně za poslední zprávu, která má stejnou prioritu.

Programy mohou buď získat první zprávu z fronty, nebo mohou získat konkrétní 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 760.

Pokud aplikace vloží posloupnost zpráv do fronty, jiná aplikace může tyto zprávy načíst ve stejném pořadí, v jakém byly vloženy, za předpokladu, že:

- Všechny zprávy mají stejnou prioritu
- Všechny zprávy byly vloženy do stejné jednotky práce, nebo všechny byly vloženy mimo jednotku práce.
- Fronta je lokální vzhledem k vkládající aplikaci.

Nejsou-li tyto podmínky splněny a aplikace závisejí na zprávách načítaných v určitém pořadí, musí aplikace před odesláním další zprávy buď zahrnout informace o řazení do dat zprávy, nebo zavést prostředky pro potvrzení přijetí zprávy.

z/OS V systému IBM MQ for z/OSmůžete použít atribut fronty *IndexType*ke zvýšení rychlosti operací MQGET ve frontě. Další informace viz “Typ indexu” na stránce 765.

Logické a fyzické řazení

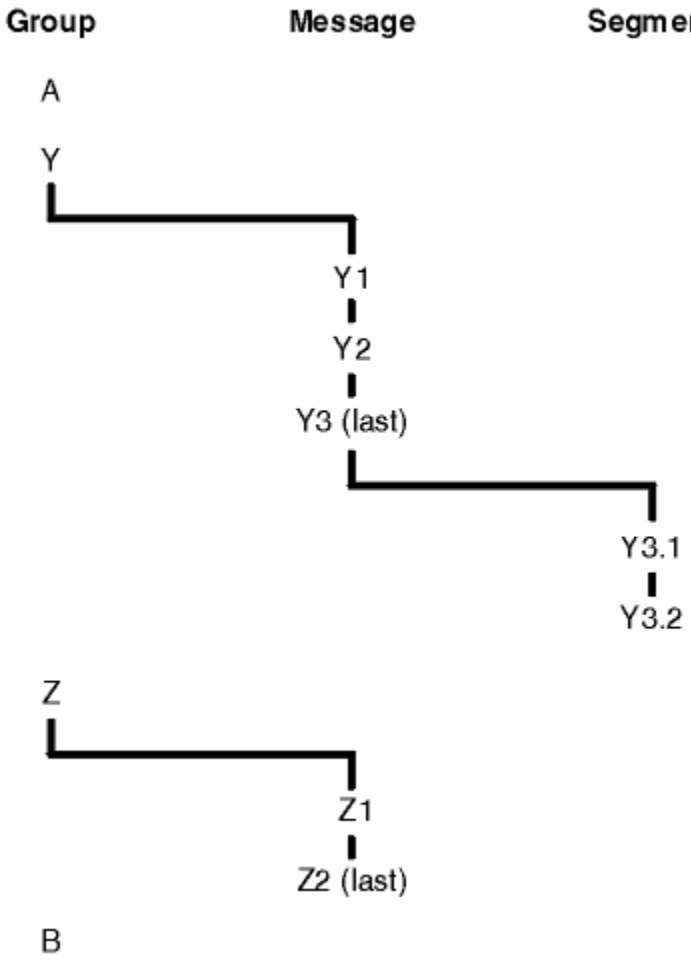
Zprávy ve frontách se mohou vyskytovat (v rámci každé úrovně priority) v *fyzickém* nebo *logickém* pořadí.

Fyzické pořadí je pořadí, ve kterém zprávy přicházejí do fronty. Logické pořadí je, když jsou všechny zprávy a segmenty ve skupině ve svém logickém pořadí, vedle sebe, v pozici určené fyzickou pozicí první položky patřící do skupiny.

Popis skupin, zpráv a segmentů viz “Skupiny zpráv” na stránce 42. Tyto fyzické a logické pořadí se mohou lišit, protože:

- Skupiny mohou do místa určení dorazit v obdobných časech z různých aplikací, a proto ztrácejí jakékoli odlišné fyzické pořadí.
- Dokonce i v rámci jedné skupiny se zprávy mohou dostat mimo pořadí kvůli přesměrování nebo prodlevě některých zpráv ve skupině.

Logické pořadí může například vypadat jako na obrázku Obrázek 61 na stránce 750:

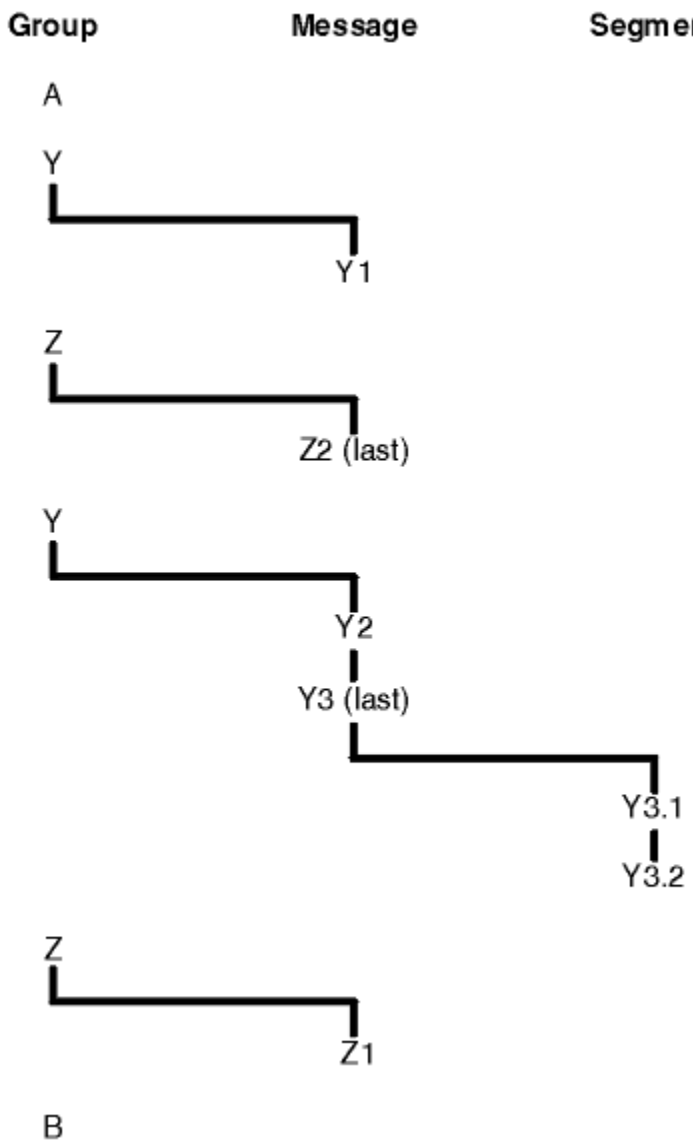


Obrázek 61. Logické pořadí ve frontě

Tyto zprávy se vyskytnou ve frontě v následujícím logickém pořadí:

1. Zpráva A (není ve skupině)
2. Logická zpráva 1 skupiny Y
3. Logická zpráva 2 skupiny Y
4. Segment 1 (poslední) logické zprávy 3 skupiny Y
5. (Poslední) segment 2 (poslední) logické zprávy 3 skupiny Y
6. Logická zpráva 1 skupiny Z
7. (Poslední) logická zpráva 2 skupiny Z
8. Zpráva B (není ve skupině)

Fyzický řád však může být zcela odlišný. Fyzická pozice *první* položky v každé skupině určuje logickou pozici celé skupiny. Pokud například skupiny Y a Z dorazily v podobných časech a zpráva 2 skupiny Z předběhla zprávu 1 ze stejné skupiny, fyzické pořadí by vypadalo jako na obrázku Obrázek 62 na stránce 751:



Obrázek 62. Fyzické pořadí ve frontě

Tyto zprávy se vyskytují ve frontě v následujícím fyzickém pořadí:

1. Zpráva A (není ve skupině)
2. Logická zpráva 1 skupiny Y
3. Logická zpráva 2 skupiny Z
4. Logická zpráva 2 skupiny Y
5. Segment 1 (poslední) logické zprávy 3 skupiny Y
6. (Poslední) segment 2 (poslední) logické zprávy 3 skupiny Y
7. Logická zpráva 1 skupiny Z
8. Zpráva B (není ve skupině)

Poznámka: V systému IBM MQ for z/OSnení fyzické pořadí zpráv ve frontě zaručeno, pokud je fronta indexována pomocí GROUPID.

Při získávání zpráv můžete zadat MQGMO_LOGICAL_ORDER, aby se zprávy načítaly v logickém pořadí, a nikoli ve fyzickém pořadí.

Pokud zadáte volání MQGET s MQGMO_BROWSE_FIRST a MQGMO_LOGICAL_ORDER, následná volání MQGET s MQGMO_BROWSE_NEXT musí také určovat MQGMO_LOGICAL_ORDER. Naopak, pokud

MQGET s MQGMO_BROWSE_FIRST neurčuje MQGMO_LOGICAL_ORDER, nesmí být následující MQGETs MQGMO_BROWSE_NEXT.

Informace o skupině a segmentu, které správce front uchovává pro volání MQGET, která procházejí zprávy ve frontě, jsou odděleny od informací o skupině a segmentu, které správce front uchovává pro volání MQGET, která odebírají zprávy z fronty. Zadáte-li MQGMO_BROWSE_FIRST, správce front bude při procházení ignorovat informace o skupině a segmentu 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, které nejsou ve skupině) bez určení MQGMO_LOGICAL_ORDER. Pokud například poslední zpráva ve skupině *předchází* první zprávu ve skupině ve frontě, použití MQGMO_BROWSE_NEXT k procházení za koncem skupiny a určení MQMO_MATCH_MSG_SEQ_NUMBER s hodnotou *MsgSeqNumber* nastavenou na 1 (k nalezení první zprávy další skupiny) vrátí první zprávu ve skupině, která již byla procházena. K tomu může dojít okamžitě, nebo několik volání MQGET později (pokud existují skupiny, které zasahují).

Vyhněte se možnosti nekonečné smyčky otevřením fronty *dvakrát* pro procházení:

- Pomocí prvního popisovače můžete procházet pouze první zprávu v každé skupině.
- Pomocí druhého popisovače můžete procházet pouze zprávy v rámci specifické skupiny.
- Pomocí voleb MQMO_ * přesuňte druhý kurzor procházení na pozici prvního kurzoru procházení před procházením zpráv ve skupině.
- Nepoužívejte procházení MQGMO_BROWSE_NEXT za koncem skupiny.

Další informace viz MQGET, MQMDa Pravidla pro ověření voleb MQI.

Pro většinu aplikací budete pravděpodobně při procházení volit buď logické, nebo fyzické řazení. Pokud však chcete přepínat mezi těmito režimy, nezapomeňte, že při prvním spuštění procházení s MQGMO_LOGICAL_ORDER je vaše pozice v logické posloupnosti vytvořena.

Pokud v tuto chvíli není přítomna první položka ve skupině, není skupina, ve které se nacházíte, považována za součást logické posloupnosti.

Jakmile je kurzor procházení ve skupině, může pokračovat ve stejné skupině, i když je první zpráva odebrána. Na začátku se však nikdy nemůžete přesunout do skupiny pomocí MQGMO_LOGICAL_ORDER, kde není první položka přítomna.

MQPMO_LOGICAL_ORDER

Volba MQPMO sděluje správci front, jak aplikace vkládá zprávy do skupin a segmentů logických zpráv. Lze jej zadat pouze pro volání MQPUT; není platný pro 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, nemusí aplikace udržovat a aktualizovat informace o skupinách a segmentech pro každé volání MQPUT, protože správce front tyto informace udržuje a aktualizuje. Konkrétně to znamená, že aplikace nemusí nastavit pole *GroupId*, *MsgSeqNumber*a *Offset* v deskriptoru MQMD, protože správce front nastaví tato pole na příslušné hodnoty. Aplikace musí nastavit pouze pole *MsgFlags* v MQMD, aby označila, kdy zprávy patří do skupin nebo jsou segmenty logických zpráv, a aby označila poslední zprávu ve skupině nebo posledním segmentu logické zprávy.

Po spuštění skupiny zpráv nebo logické zprávy musí následná volání MQPUT určit příslušné příznaky MQMF_ * v souboru *MsgFlags* v deskriptoru MQMD. Pokud se aplikace pokusí vložit zprávu, která není

ve skupině, když existuje neukončená skupina zpráv, nebo vložit zprávu, která není segmentem, když existuje neukončená logická zpráva, volání selže s kódem příčiny MQRC_INCOMPLETE_GROUP nebo MQRC_INCOMPLETE_MSG, podle potřeby. Správce front však uchovává informace o aktuální skupině zpráv nebo aktuální logické zprávě a aplikace je může ukončit odesláním zprávy (pravděpodobně bez dat zprávy aplikace) s uvedením MQMF_LAST_MSG_IN_IN_GROUP nebo MQMF_LAST_SEGMENT podle potřeby před opětovným zadáním volání MQPUT pro vložení zprávy, která není ve skupině nebo není segmentem.

Obrázek 62 na stránce 751 zobrazuje platné kombinace voleb a příznaků a hodnoty polí *GroupId*, *MsgSeqNumber*a *Offset* , které správce front používá v jednotlivých případech. Kombinace voleb a příznaků, které nejsou zobrazeny v tabulce, nejsou platné. Sloupce v tabulce mají následující význam: Buď znamená Ano, nebo Ne:

PROTOKOLNÍ ZÁZNAM
Určuje, zda je pro volání určena volba MQPMO_LOGICAL_ORDER.

MIG
Zda je ve volání zadána volba MQMF_MSG_IN_GROUP nebo MQMF_LAST_MSG_IN_GROUP.

Skupina SEG
Určuje, zda je ve volání zadána volba MQMF_SEGMENT nebo MQMF_LAST_SEGMENT.

SEG v pořádku
Určuje, zda je ve volání zadána volba MQMF_SEGMENTATION_ALLOWED.

Aktuální grp
Zda před voláním existuje aktuální skupina zpráv.

Aktuální zpráva protokolu
Zda před voláním existuje aktuální logická zpráva.

Ostatní sloupce
Zobrazit hodnoty, které správce front používá. Předchozí označuje hodnotu použitou pro pole v předchozí zprávě pro popisovač fronty.

Tabulka 116. Volby MQPUT související se zprávami ve skupinách a segmentech logických zpráv

Volby, které zadáte	Volby, které zadáte	Volby, které zadáte	Volby, které zadáte	Stav skupiny a zprávy protokolu před voláním	Stav skupiny a zprávy protokolu před voláním	Hodnoty, které správce front používá	Hodnoty, které správce front používá	Hodnoty, které správce front používá
PROTOKOLNÍ ZÁZNAM	MIG	Skupina SEG	SEG v pořádku	Aktuální grp	Aktuální zpráva protokolu	**GroupId**	**MsgSeqNumber**	**Offset**
Ano	Ne	Ne	Ne	Ne	Ne	MQGI_NONE	1	0
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	ID předchozí skupiny	1	Předchozí offset + předchozí délka segmentu
Ano	Ano	buď	buď	Ne	Ne	ID nové skupiny	1	0

Tabulka 116. Volby MQPUT související se zprávami ve skupinách a segmentech logických zpráv (pokračování)

Volby, které zadáte	Volby, které zadáte	Volby, které zadáte	Volby, které zadáte	Stav skupiny a zprávy protokolu před voláním	Stav skupiny a zprávy protokolu před voláním	Hodnoty, které správce front používá	Hodnoty, které správce front používá	Hodnoty, které správce front používá
Ano	Ano	buď	buď	Ano	Ne	ID předchozí skupiny	Předchozí pořadové číslo + 1	0
Ano	Ano	Ano	buď	Ano	Ano	ID předchozí skupiny	Předchozí pořadové číslo	Předchozí offset + předchozí délka segmentu
Ne	Ne	Ne	Ne	buď	buď	MQGI_NONE	1	0
Ne	Ne	Ne	Ano	buď	buď	Nové ID skupiny, pokud je MQGI_NONE, jinak hodnota v poli	1	0
Ne	Ne	Ano	buď	buď	buď	Nové ID skupiny, pokud je MQGI_NONE, jinak hodnota v poli	1	Hodnota v poli
Ne	Ano	Ne	buď	buď	buď	Nové ID skupiny, pokud je MQGI_NONE, jinak hodnota v poli	Hodnota v poli	0
Ne	Ano	Ano	buď	buď	buď	Nové ID skupiny, pokud je MQGI_NONE, jinak hodnota v poli	Hodnota v poli	Hodnota v poli

Poznámka:

- Parametr MQPMO_LOGICAL_ORDER není platný pro volání MQPUT1 .
- Pro pole *MsgId* vygeneruje správce front nový identifikátor zprávy, pokud je zadáno MQPMO_NEW_MSG_ID nebo MQMI_NONE, a jinak použije hodnotu v poli.
- Pro pole *CorrelId* generuje správce front nový identifikátor korelace, pokud je zadán parametr MQPMO_NEW_CORREL_ID, a jinak použije hodnotu v poli.

Zadáte-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 deskriptoru MQMD, tj. všechny musí být trvalé nebo všechny musí být dočasné. Není-li tato podmínka splněna, volání MQPUT se nezdaří s kódem příčiny MQRC_INCONSISTENT_PERSISTENCE.

Volba MQPMO_LOGICAL_ORDER ovlivňuje pracovní jednotky 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 logické zprávě vloženy do pracovní jednotky, pokud je použit stejný manipulátor fronty. Nemusí však být vloženy do stejné pracovní jednotky, což umožňuje rozdělit skupinu zpráv nebo logickou zprávu, která se skládá z mnoha fyzických zpráv, na dvě nebo více po sobě jdoucích pracovních jednotek pro popisovač fronty.

- Není-li první fyzická zpráva ve skupině nebo logické zprávě vložena do pracovní jednotky, nemůže být žádná z ostatních fyzických zpráv ve skupině nebo logické zprávě vložena do pracovní jednotky, pokud je použita stejná obsluha fronty.

Nejsou-li tyto podmínky splněny, volání MQPUT selže s kódem příčiny MQRC_INCONSISTENT_UOW.

Je-li zadána volba MQPMO_LOGICAL_ORDER, nesmí být hodnota MQMD zadaná ve volání MQPUT menší než hodnota MQMD_VERSION_2. Není-li tato podmínka splněna, volání selže s kódem příčiny MQRC_WRONG_MD_VERSION.

Není-li zadán parametr MQPMO_LOGICAL_ORDER, lze zprávy ve skupinách a segmentech logických zpráv vkládat v libovolném pořadí a není nutné vkládat úplné skupiny zpráv ani úplné logické zprávy. Je odpovědností aplikace zajistit, aby pole *GroupId*, *MsgSeqNumber*, *Offset*a *MsgFlags* měla odpovídající hodnoty.

Tuto techniku použijte k restartování skupiny zpráv nebo logické zprávy uprostřed po selhání systému. Když se systém restartuje, aplikace může nastavit pole *GroupId*, *MsgSeqNumber*, *Offset*, *MsgFlags*a *Persistence* na odpovídající hodnoty a pak zadat volání MQPUT s nastavením MQPMO_SYNCPOINT nebo MQPMO_NO_SYNCPOINT podle potřeby, ale bez uvedení MQPMO_LOGICAL_ORDER. Je-li toto volání úspěšné, správce front zachová informace o skupině a segmentu a následná volání MQPUT používající tento manipulátor fronty mohou určit MQPMO_LOGICAL_ORDER jako normální.

Informace o skupině a segmentu, které správce front uchovává pro volání MQPUT, jsou odděleny od informací o skupině a segmentu, které uchovává pro volání MQGET.

Pro každý daný manipulátor fronty může aplikace kombinovat volání MQPUT, která určují MQPMO_LOGICAL_ORDER, s voláními MQPUT, která nikoli, ale všimněte si následujících bodů:

- Pokud není zadána hodnota MQPMO_LOGICAL_ORDER, každé úspěšné volání MQPUT způsobí, že správce front nastaví informace o skupině a segmentu pro manipulátor fronty na hodnoty určené aplikací a nahradí existující informace o skupině a segmentu uchovávané správcem front pro manipulátor fronty.
- Není-li zadán parametr MQPMO_LOGICAL_ORDER, volání neselže, pokud existuje aktuální skupina zpráv nebo logická zpráva; volání může být úspěšné s kódem dokončení MQCC_WARNING. Tabulka 117 na stránce 755 zobrazuje různé případy, které mohou nastat. V těchto případech, pokud kód dokončení není MQCC_OK, je kód příčiny jeden z následujících (podle potřeby):
  - MQRC_INCOMPLETE_GROUP
  - MQRC_INCOMPLETE_MSG
  - MQRC_NEKONZISTENTNÍ_PERZISTENCE
  - MQRC_INCONSISTENT_UOW

  Poznámka: Správce front nekontroluje informace o skupině a segmentu pro volání MQPUT1 .

Tabulka 117. 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	MQCC_FAILED	MQCC_FAILED
MQPUT bez MQPMO_LOGICAL_ORDER	MQCC_VAROVÁNÍ	MQCC_OK
MQCLOSE s neukončenou skupinou nebo logickou zprávou	MQCC_VAROVÁNÍ	MQCC_OK

Pro aplikace, které vkládají zprávy a segmenty v logickém pořadí, zadejte MQPMO_LOGICAL_ORDER, protože se jedná o nejjednodušší volbu, která se má použít. Tato volba zbavuje aplikaci

potřeby spravovat informace o skupině a segmentu, protože správce front tyto informace spravuje. Specializované aplikace však mohou vyžadovat větší kontrolu, než jakou poskytuje volba MQPMO_LOGICAL_ORDER, čehož lze dosáhnout neurčením této volby. Pokud tak učiníte, musíte se před každým voláním MQPUT nebo MQPUT1 ujistit, že jsou správně nastavena pole *GroupId*, *MsgSeqNumber*, *Offset*a *MsgFlags* .

Například aplikace, která chce předávat přijaté fyzické zprávy bez ohledu na to, zda jsou tyto zprávy ve skupinách nebo segmentech logických zpráv, nesmí určovat MQPMO_LOGICAL_ORDER, a to ze dvou důvodů:

- Pokud jsou zprávy načítány a řazeny do pořadí, zadáním MQPMO_LOGICAL_ORDER přiřadíte ke zprávám nový identifikátor skupiny, který může původci zpráv ztížit nebo znemožnit korelaci odpovědí nebo zpráv sestav, které jsou výsledkem skupiny zpráv.
- V komplexní síti s více cestami mezi odesílajícími a přijímajícími správci front mohou fyzické zprávy přicházet mimo pořadí. Nezadáte-li ve volání MQGET parametry MQPMO_LOGICAL_ORDER a MQGMO_LOGICAL_ORDER, může předávající aplikace načíst a předat každou fyzickou zprávu ihned po jejím doručení, aniž by čekala na další zprávu v logickém pořadí.

Aplikace, které generují zprávy sestavy pro zprávy ve skupinách nebo segmentech logických zpráv, také nesmí při vkládání zprávy sestavy určovat MQPMO_LOGICAL_ORDER.

MQPMO_LOGICAL_ORDER lze zadat s jakoukoli jinou volbou MQPMO_ *.

Vložení 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, budou směrovány na jedinou instanci. To má nevýhodu, že provoz aplikace není vyvažován v rámci více instancí fronty klastru. Chcete-li povolit vyrovnávání pracovní zátěže při zachování neporušenosti skupin zpráv, musíte nastavit následující volby:

- Volání MQPUT musí určovat MQPMO_LOGICAL_ORDER
- Volání MQOPEN musí určovat jednu z následujících dvou voleb:
  - MQOO_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 požadavků na MQCLOSE a MQOPEN fronty. *Mezi skupinami* znamená, že proměnná MQMF_MSG_IN_GROUP je nastavena v deskriptoru MQMD (v2) nebo MQMDE a že neprobíhá žádná částečně dokončená skupina. Když probíhá zpracování skupiny, znovu se použije vyřešený správce front a název fronty v popisovači objektu.

Pokud byla předchozí zpráva MQPMO_LOGICAL_ORDER a/nebo MQMF_MSG_IN_GROUP nastavena, ale aktuální zpráva není součástí skupiny, volání PUT se nezdaří s hodnotou MQRC_INCOMPLETE_GROUP.

Pokud jednotlivá operace MQPUT neurčuje hodnotu MQPMO_LOGICAL_ORDER a není aktivní žádná aktuální skupina, je pro tuto zprávu řízeno vyrovnávání pracovní zátěže (jako by volání MQOPEN určilo hodnotu MQOOO_BIND_NOT_FIXED).

Pro zprávy svázané s místem určení pomocí MQOO_BIND_ON_GROUP není provedeno žádné opětovné přidělení. Další informace o realokaci viz “Skupiny zpráv” na stránce 42.

Seskupení logických zpráv

Existují dvě hlavní příčiny použití logických zpráv ve skupině:

- Možná budete muset zpracovat zprávy v určitém pořadí.
- Možná budete muset zpracovat každou zprávu ve skupině souvisejícím způsobem.

V obou případech načtěte celou skupinu se stejnou získávajícími instancemi aplikace.

Předpokládejme například, že se skupina skládá ze čtyř logických zpráv. Vkládající aplikace vypadá 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 provádějící získávání určuje volbu MQGMO_ALL_MSGS_AVAILABLE pro první zprávu ve skupině. Tím se zajistí, že zpracování se nespustí, dokud nedorazí všechny zprávy v rámci skupiny. Volba MQGMO_ALL_MSGS_AVAILABLE je pro následné zprávy v rámci skupiny ignorována.

Při načtení první logické zprávy skupiny můžete pomocí příkazu MQGMO_LOGICAL_ORDER zajistit, aby byly zbývající logické zprávy skupiny načteny v pořadí.

Takže, získání aplikace 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 aplikací logických zpráv” na stránce 768 a “Uvedení a získání skupiny, která zahrnuje jednotky práce” na stránce 757.

Upozornění: Při použití publikování/odběru k odesílání zpráv do tématu (nebo vkládání zpráv do aliasu tématu) není seskupení zpráv a segmentace povolena.

Vzhledem k tomu, že odběry lze vytvářet a odebírat nezávisle na aktivitě publikování, nelze zajistit, aby odběratel obdržel úplnou skupinu zpráv nebo všechny segmenty zprávy; viz RC2417: MQRC_MSG_NOT_ALLOWED_IN_GROUP.

Chcete-li získat informace o tom, jak povolit aplikaci požadovat, aby byla skupina zpráv přidělena stejné cílové instanci pro fronty klastru, prohlédněte si téma DefBind.

Uvedení a získání skupiny, která zahrnuje jednotky práce

V předchozím případě nemohou zprávy nebo segmenty začít opouštět uzel (pokud je jeho cíl vzdálený) nebo začít načítat, dokud nebude vložena celá skupina a dokud nebude potvrzena jednotka práce. To nemusí být to, co chcete, pokud vložení celé skupiny trvá dlouho, nebo pokud je prostor fronty na uzlu omezený. Chcete-li to překonat, dejte skupinu do několika pracovních jednotek.

Pokud je skupina vložena do více pracovních jednotek, je možné, aby se část skupiny odevzdala i v případě, že se vkládající aplikace nezdaří. Aplikace proto musí uložit informace o stavu, potvrzené s každou pracovní jednotkou, kterou může použít po restartu k obnovení neúplné skupiny. Nejjednodušší místo pro záznam těchto informací je ve frontě STATUS. Pokud byla úplná skupina úspěšně vložena, fronta STATUS je prázdná.

Pokud se jedná o segmentaci, logika je podobná. V tomto případě musí soubor **StatusInfo** obsahovat soubor *Offset*.

Zde je příklad vložení skupiny do několika pracovních jednotek:

```
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 všechny jednotky práce potvrzeny, celá skupina byla úspěšně vložena a fronta STATUS je prázdná. Pokud ne, musí být skupina obnovena v bodě označeném informacemi o stavu. MQPMO_LOGICAL_ORDER nelze použít pro první vložení, ale může poté.

Zpracování restartu 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 chtít začít zpracovávat zprávy ve skupině před tím, než dorazí celá skupina. To zlepšuje dobu odezvy zpráv ve skupině a také znamená, že úložiště není vyžadováno pro celou skupinu. Chcete-li si uvědomit výhody, použijte několik pracovních jednotek pro každou skupinu zpráv. Z důvodů obnovy musíte načíst každou zprávu v rámci pracovní jednotky.

Stejně jako u odpovídající vkládající aplikace to vyžaduje, aby byly informace o stavu zaznamenány někde automaticky, když je každá jednotka práce potvrzena. Opět platí, že nejjednodušší místo pro záznam těchto informací je ve frontě STATUS. Pokud byla úplná skupina úspěšně zpracována, fronta STATUS je prázdná.

Poznámka: V případě mezilehlých pracovních jednotek se můžete vyhnout voláním MQGET z fronty STATUS tím, že určíte, že každý MQPUT do stavové fronty je segmentem zprávy (tj. nastavením příznaku MQMF_SEGMENT), místo abyste pro každou pracovní jednotku vložili úplnou novou zprávu. V poslední pracovní jednotce je konečný segment vložen do stavové fronty s uvedením MQMF_LAST_SEGMENT a pak jsou informace o stavu vymazány s MQGET určujícím MQGMO_COMPLETE_MSG.

Během zpracování restartu namísto použití jediného příkazu MQGET pro získání možné stavové zprávy procházejte stavovou frontu s MQGMO_LOGICAL_ORDER, dokud nedosáhnete posledního segmentu (tj. dokud nebudou vráceny žádné další segmenty). V první pracovní jednotce po restartu uveďte také offset explicitně při vkládání stavového segmentu.

V následujícím příkladu považujeme pouze zprávy v rámci skupiny za předpokladu, že vyrovnávací paměť aplikace je vždy dostatečně velká na to, aby zadržela celou zprávu, bez ohledu na to, zda byla zpráva segmentována. Hodnota MQGMO_COMPLETE_MSG je proto určena pro každý příkaz MQGET. Stejné zásady platí, pokud se jedná o segmentaci (v tomto případě musí StatusInfo obsahovat *Offset*).

Pro jednoduchost předpokládáme, že v rámci jedné jednotky UOW jsou načteny maximálně 4 zprávy:

```
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 všechny pracovní jednotky potvrzeny, byla úspěšně načtena celá skupina a fronta STATUS je prázdná. Pokud ne, musí být skupina obnovena v bodě označeném informacemi o stavu. MQGMO_LOGICAL_ORDER nelze použít pro první načtení, ale může poté.

Zpracování restartu 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 řada způsobů, jak získat konkrétní zprávu z fronty. Jedná se o: výběr položek MsgId a CorrelId, výběr položek GroupId, MsgSeqNumber a Offset a výběr položek MsgToken. Při otevírání fronty můžete také použít řetězec výběru.

Chcete-li získat konkrétní zprávu z fronty, použijte pole `MsgId` a `CorrelId` struktury MQMD. Aplikace však mohou tato pole explicitně nastavit, takže zadané hodnoty nemusí identifikovat jedinečnou zprávu. Tabulka 118 na stránce 760 zobrazuje, která zpráva je načtena pro možná nastavení těchto polí. Tato pole jsou na vstupu ignorována, pokud zadáte parametr MQGMO_MSG_UNDER_CURSOR v parametru **`GetMsgOpts`** volání MQGET.

Tabulka 118. Použití identifikátorů zprávy a korelace

Chcete-li načíst ...	**`MsgId`**	**`CorrelId`**
První zpráva ve frontě	`MQMI_NONE`	`MQCI_NONE`
První zpráva, která odpovídá *`MsgId`*	Nenulová	`MQCI_NONE`
První zpráva, která odpovídá *`CorrelId`*	`MQMI_NONE`	Nenulová
První zpráva, která odpovídá *`MsgId`* i *`CorrelId`*	Nenulová	Nenulová

V každém případě *první* znamená první zprávu, která splňuje kritéria výběru (není-li zadána hodnota MQGMO_BROWSE_NEXT, znamená-li *další* zprávu v posloupnosti splňující kritéria výběru).

Při návratu volání MQGET nastaví pole *`MsgId`* a *`CorrelId`* na identifikátory zprávy a korelace vrácené zprávy, pokud existují.

Nastavíte-li pole *`Version`* struktury MQMD na hodnotu 2, můžete použít pole *`GroupId`*, *`MsgSeqNumber`*a *`Offset`* . Tabulka 119 na stránce 760 zobrazuje, která zpráva je načtena pro možná nastavení těchto polí.

Tabulka 119. Použití identifikátoru skupiny

Chcete-li načíst ...	**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_OFFSET

Notes:

1. MQMO_MATCH_XXX znamená, že pole *XXX* ve struktuře MQMD je nastaveno na hodnotu, která se má shodovat.
2. Příznaky MQMO lze použít v kombinaci. Například 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. Zadáte-li MQGMO_LOGICAL_ORDER, bude ovlivněna zpráva, kterou se pokoušíte načíst, protože tato volba závisí na informacích o stavu řízených pro manipulátor fronty. Informace o tomto tématu naleznete v části “Logické a fyzické řazení” na stránce 749 a Volby.

Volání MQGET obvykle načte první zprávu z fronty. Zadáte-li konkrétní zprávu při použití volání MQGET, musí správce front prohledávat frontu, dokud tuto zprávu nenajde. To může ovlivnit výkon vaší aplikace.

Pokud používáte verzi 2 nebo novější struktury MQGMO a neurčujete příznaky MQMO_MATCH_MSG_ID nebo MQMO_MATCH_CORREL_ID, nemusíte resetovat pole `MsgId` nebo `CorrelId` mezi příkazy MQGETs.

z/OS V systému IBM MQ for z/OSlze atribut fronty `IndexType` použít ke zvýšení rychlosti operací MQGET ve frontě. Další informace viz téma “Typ indexu” na stránce 765.

Specifickou zprávu můžete získat z fronty zadáním jejího MsgToken a MatchOption MQMO_MATCH_MSG_TOKEN ve struktuře MQGMO. MsgToken je vrácen voláním MQPUT, které původně vložilo tuto zprávu do fronty, nebo předchozími operacemi MQGET a zůstává konstantní, pokud není správce front restartován.

Máte-li zájem pouze o podmnožinu zpráv ve frontě, můžete určit, které zprávy chcete zpracovat, pomocí řetězce výběru s voláním MQOPEN nebo MQSUB. Příkaz MQGET poté načte další zprávu, která splňuje daný řetězec výběru. Další informace o výběrových řetězcích viz “Selektory.” na stránce 30.

Zlepšení výkonu dočasných zpráv

Když klient vyžaduje zprávu ze serveru, odešle požadavek na server. Odešle samostatný požadavek pro každou zprávu, kterou spotřebuje. Chcete-li zlepšit výkon klienta spotřebovávajícího přechodné zprávy tím, že se vyhnete odesílání těchto zpráv požadavků, lze klienta nakonfigurovat tak, aby používal *dopředné čtení*. Dopředné čtení umožňuje, aby byly zprávy odesílány klientovi, aniž by je aplikace musela požadovat.

Je-li povoleno dopředné čtení, jsou zprávy odesílány do vyrovnávací paměti na klientovi s názvem *vyrovnávací paměť dopředného čtení*. Klient bude mít vyrovnávací paměť dopředného čtení pro každou frontu, kterou má otevřenou, s povoleným dopředným čtením. Zprávy ve vyrovnávací paměti dopředného čtení nejsou trvalé. Klient pravidelně aktualizuje server s 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:

- 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 příjmu dočasných zpráv z klientské aplikace. Toto zlepšení výkonu je k dispozici pro aplikace MQI i JMS . Klientské aplikace používající příkaz MQGET nebo asynchronní spotřebu budou mít prospěch ze zlepšení výkonu při spotřebě dočasných zpráv.

Ne všechny návrhy aplikací klienta jsou vhodné pro použití dopředného čtení, protože ne všechny volby jsou podporovány pro použití s dopředným čtením a některé volby musí být konzistentní mezi voláními MQGET, když je povoleno dopředné čtení. Pokud klient změní svá kritéria výběru mezi voláními MQGET, zprávy uložené ve vyrovnávací paměti dopředného čtení zůstanou ve vyrovnávací paměti dopředného čtení klienta.

Pokud již nejsou vyžadovány nevyřízené zprávy s předchozími výběrovými kritérii, lze na klientovi nastavit konfigurovatelný interval mazání, aby se tyto zprávy z klienta automaticky vyprazdňovaly. Interval vyprazdňování je jednou ze skupin voleb vyladění dopředného čtení určených klientem. Tyto možnosti je možné vyladit tak, aby splňovaly vaše požadavky.

Pokud je aplikace klienta restartována, zprávy ve vyrovnávací paměti dopředného čtení mohou být ztraceny. Naopak zpráva, která byla přesunuta do vyrovnávací paměti dopředného čtení, může být odstraněna ze základní fronty; to nevede k jejímu odebrání z vyrovnávací paměti, takže volání MQGET používající dopředné čtení může vrátit zprávu, která již neexistuje.

Dopředné čtení se provádí pouze pro vazby klienta. Atribut je ignorován pro všechny ostatní vazby.

Dopředné čtení nemá žádný vliv na spouštění. Při dopředném čtení zprávy klientem se negeneruje žádná zpráva spouštěče. Dopředné čtení negeneruje informace o evidenci a statistice, je-li povoleno.

Použití dopředného čtení s publikovaným systémem zpráv odběru

Když odebírající aplikace uvádí cílovou frontu, do které jsou odesílána publikování, použije se hodnota DEFREADA uvedené fronty jako výchozí hodnota dopředného čtení.

Když odebírající aplikace požaduje, aby produkt IBM MQ spravoval místo určení, kam jsou odesílána publikování, vytvoří se spravovaná fronta jako dynamická fronta založená na předdefinované modelové frontě. 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 používají, pokud není pro toto nebo nadřízené téma definována modelová fronta.

Související pojmy

"Vyladění výkonu pro přechodné zprávy v systému AIX" na stránce 764
Používáte-li produkt AIX V5.3 nebo novější, zvažte nastavení parametru ladění tak, aby pro přechodné zprávy používal úplný výkon.

Související úlohy

"Povolení a zakázání dopředného čtení" na stránce 763
Při výchozím nastavení je dopředné čtení zakázáno. Čtení napřed můžete povolit na úrovni fronty nebo aplikace.

Související odkazy

"Volby MQGET a dopředné čtení" na stránce 762
Při povolení dopředného čtení nejsou podporovány všechny volby MQGET. Některé volby musí být mezi voláními MQGET konzistentní.

Volby MQGET a dopředné čtení

Při povolení dopředného čtení nejsou podporovány všechny volby MQGET. Některé volby musí být mezi voláními MQGET konzistentní.

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:

- 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 je lze změnit mezi voláními MQGET.

Tabulka 120. Volby MQGET a dopředné čtení

Hodnoty a volby MQGET	**Povoleno, když je povoleno dopředné čtení a lze jej změnit mezi voláními MQGET** [5]	**Povoleno, když je povoleno dopředné čtení, ale nelze je měnit mezi voláními MQGET** [1]	**Volby MQGET, které nejsou povoleny, když je povoleno dopředné čtení** [2]
Hodnoty MQGET MQMD	MsgId [3] CorrelId [3]	Kódování CodedCharSetId	

Tabulka 120. Volby MQGET a dopředné čtení (pokračování)			
Hodnoty a volby MQGET	**Povoleno, když je povoleno dopředné čtení a lze jej změnit mezi voláními MQGET [5]**	**Povoleno, když je povoleno dopředné čtení, ale nelze je měnit mezi voláními MQGET [1]**	**Volby MQGET, které nejsou povoleny, když je povoleno dopředné čtení [2]**
Volby MQGET MQGMO	• MQGMO_NO_WAIT • MQGMO_BROWSE_MESSAGE _UNDER_CURSOR • MQGMO_BROWSE_FIRST • MQGMO_BROWSE_NEXT • MQGMO_FAIL_IF_QUIESCING	• MQGMO_SYNCPOINT_IF _PERSISTENT • MQGMO_NO_SYNCPOINT • MQGMO_ACCEPT_TRUNCATED _MSG • MQGMO_CONVERT	• MQGMO_SET_SIGNAL • MQGMO_SYNCPOINT • MQGMO_MARK_SKIP _BACKOUT • MQGMO_MSG_UNDER _CURSOR [4] • MQGMO_LOCK • MQGMO_UNLOCK • MQGMO_LOGICAL_ORDER • MQGMO_COMPLETE_MSG • MQGMO_ALL_MSGS_AVAILABLE • MQGMO_ALL_SEGMENTS_ K DISPOZICI

Notes:

1. Pokud jsou tyto volby změněny mezi voláními MQGET, vrátí se 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 dopředného čtení klienta, dokud nebudou spotřebovány (nebo automaticky vyprázdněny).
4. MQGMO_MSG_UNDER_CURSOR nelze použít s dopředným čtením. Dopředné čtení je zakázáno, pokud jsou při otevírání fronty zadány volby MQOO_BROWSE a MQOO_INPUT_SHARED nebo MQOO_INPUT_EXCLUSIVE.
5. Je-li povoleno dopředné čtení, první příkaz MQGET určí, zda mají být zprávy procházeny nebo získávány z fronty. Pokud aplikace klienta poté použije příkaz MQGET se změněnými volbami, jako je například pokus o procházení po počátečním načtení nebo pokus o získání po počátečním procházení, vrátí se kód příčiny MQRC_OPTIONS_CHANGED.

Pokud klient změní kritéria výběru mezi voláními MQGET, zprávy uložené ve vyrovnávací paměti dopředného čtení, které odpovídají počátečním kritériím výběru, nejsou aplikací klienta spotřebovány a zůstávají ve vyrovnávací paměti dopředného čtení klienta. V situacích, kdy vyrovnávací paměť dopředného čtení klienta obsahuje mnoho uvízlých zpráv, jsou výhody spojené s dopředným čtením ztraceny a pro každou spotřebovanou zprávu je vyžadován samostatný požadavek na server. Chcete-li určit, zda je dopředné čtení efektivně používáno, můžete použít parametr stavu připojení READA.

Dopředné čtení může být na žádost aplikace zablokováno kvůli nekompatibilním volbám uvedeným v prvním volání MQGET. V této situaci stav připojení ukazuje dopředné čtení jako blokováno.

Pokud se kvůli těmto omezením pro příkaz MQGET rozhodnete, že návrh aplikace klienta není vhodný pro dopředné čtení, zadejte volbu MQOPEN MQOO_READ_AHEAD_NO. Případně nastavte výchozí hodnotu dopředného čtení fronty, která se otevírá, na hodnotu NO nebo DISABLED.

Povolení a zakázání dopředného čtení
Při výchozím nastavení je dopředné čtení zakázáno. 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:

- 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.

Chcete-li povolit dopředné čtení:

- Chcete-li konfigurovat dopředné čtení na úrovni fronty, nastavte atribut fronty, DEFREADA na hodnotu YES.
- Chcete-li konfigurovat dopředné čtení na úrovni aplikace, postupujte takto:
  - pro použití dopředného čtení, kdykoli je to možné, použijte volbu MQOO_READ_AHEAD ve volání funkce MQOPEN. Není možné, aby aplikace klienta používala dopředné čtení, pokud byl atribut fronty DEFREADA nastaven na hodnotu DISABLED.
  - Chcete-li používat dopředné čtení pouze v případě, že je ve frontě povoleno dopředné čtení, použijte volbu MQO_READ_AHEAD_AS_Q_DEF pro volání funkce MQOPEN.

Pokud návrh klientské aplikace není vhodný pro dopředné čtení, můžete jej zakázat:

- na úrovni fronty nastavením atributu fronty DEFREADA na hodnotu NO, pokud nechcete, aby se používalo dopředné čtení, pokud to nevyžaduje klientská aplikace, nebo na hodnotu DISABLED, pokud nechcete, aby se dopředné čtení používalo bez ohledu na to, zda je dopředné čtení vyžadováno klientskou aplikací.
- na úrovni aplikace pomocí volby MQOO_NO_READ_AHEAD volání funkce MQOPEN.

Dvě volby MQCLOSE vám umožňují konfigurovat, co se stane se zprávami, které jsou uloženy ve vyrovnávací paměti dopředného čtení, pokud je fronta zavřená.

- Pomocí příkazu MQCO_IMMEDIATE zrušte zprávy ve vyrovnávací paměti dopředného čtení.
- Použijte MQCO_QUIESCE, abyste zajistili, že zprávy ve vyrovnávací paměti dopředného čtení budou spotřebovány aplikací před uzavřením fronty. Je-li vydán příkaz MQCLOSE s hodnotou MQCO_QUIESCE a ve vyrovnávací paměti pro dopředné čtení zbývají zprávy, vrátí parametr MQRC_READ_AHEAD_MSGS hodnotu MQCC_WARNING.

Vyladění výkonu pro přechodné zprávy v systému AIX
Používáte-li produkt AIX V5.3 nebo novější, zvažte nastavení parametru ladění tak, aby pro přechodné zprávy používal úplný výkon.

Chcete-li nastavit parametr ladění tak, aby se projevoval okamžitě, zadejte jako uživatel root následující příkaz:

```
/usr/sbin/ioo -o j2_nPagesPerWriteBehindCluster=0
```

Chcete-li nastavit parametr ladění tak, aby se projevil okamžitě a přetrvá po opětovném zavedení systému, zadejte následující příkaz jako uživatel root:

```
/usr/sbin/ioo -p -o j2_nPagesPerWriteBehindCluster=0
```

Obvykle jsou přechodné zprávy uchovávány pouze v paměti, ale existují okolnosti, kdy může produkt AIX naplánovat přechodné zprávy pro zápis na disk. Zprávy, jejichž zápis na disk je naplánován, nejsou k dispozici pro příkaz MQGET, dokud nebude zápis na disk dokončen. Navrhovaný příkaz pro vyladění tuto prahovou hodnotu mění; místo plánování zápisu zpráv na disk při zařazení 16 kilobajtů dat do fronty se zápis na disk vyskytne pouze v případě, že se reálné úložiště na počítači přiblíží k zaplnění. Jedná se o globální změnu a může ovlivnit další softwarové komponenty.

V systému AIXpři použití aplikací s podporou podprocesů a zejména při spouštění na počítačích s více procesory důrazně doporučujeme před spuštěním aplikace nastavit AIXTHREAD_SCOPE=S v ID

mqm `.profile` nebo nastavit AIXTHREAD_SCOPE=S v prostředí pro lepší výkon a pevnější plánování. Příklad:

```
export AIXTHREAD_SCOPE=S
```

Nastavení AIXTHREAD_SCOPE=S znamená, že uživatelské podprocesy vytvořené s výchozími atributy jsou umístěny do rozsahu soupeření v rámci celého systému. Pokud je uživatelský podproces vytvořen s rozsahem soupeření v rámci 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 procesu s podporou podprocesů, jako je například proces agenta, můžete dosáhnout měkkého limitu pro deskriptory souborů. Toto omezení vám poskytuje IBM MQ kód příčiny `MQRC_UNEXPECTED_ERROR (2195)` a, pokud je k dispozici dostatek deskriptorů souboru, soubor IBM MQ FFST™ .

Chcete-li se vyhnout tomuto problému, můžete zvýšit limit procesu pro počet deskriptorů souboru. Chcete-li tak učinit, změňte atribut `nofiles` v souboru `/etc/security/limits` na hodnotu 10.000 pro ID uživatele mqm nebo ve výchozí sekci.

Omezení systémových prostředků

Nastavte limit systémových prostředků pro datový segment a segment zásobníku na neomezenou hodnotu pomocí následujících příkazů v příkazovém řádku:

```
ulimit -d unlimited
ulimit -s unlimited
```

Typ indexu

Atribut fronty *IndexType*určuje typ indexu, který správce front udržuje, aby zvýšil rychlost operací MQGET ve frontě.

Poznámka: Podporováno pouze na systému 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 749).
groupID	Je udržován index identifikátorů skupin. Tento typ indexu musíte použít, chcete-li logické řazení skupin zpráv (viz “Logické a fyzické řazení” na stránce 749).
MSGID	Je udržován index identifikátorů zpráv. Tuto volbu použijte při načítání zpráv s použitím pole *MsgId* jako kritéria výběru pro volání MQGET (viz “Získání konkrétní zprávy” na stránce 760).
MsgToken	Je udržován index tokenů zpráv.
CorrelId	Je udržován index identifikátorů korelace. Tuto volbu použijte při načítání zpráv s použitím pole *CorrelId* jako kritéria výběru pro volání MQGET (viz “Získání konkrétní zprávy” na stránce 760).

Poznámka:

1. Pokud indexujete pomocí volby MSGID nebo CORRELID, nastavte relativní parametry **MsgId** nebo **CorrelId** v deskriptoru MQMD. Není výhodné nastavit obojí.
2. Procházení používá mechanizmus indexování k vyhledání zprávy, pokud fronta splňuje všechny následující podmínky:

- Má typ indexu MSGID, CORRELID nebo GROUPID
- Prohlížel se se stejným typem ID
- Má zprávy pouze s jednou prioritou

3. Vyhněte se frontám (indexovaným pomocí *MsgId* nebo *CorrelId*), které obsahují tisíce zpráv, protože to ovlivňuje čas restartu. (Toto 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** viz IndexType. Další informace o atributu **IndexType** viz “Aspekty návrhu a výkonu pro aplikace z/OS” na stránce 62.

Zpracování zpráv delších než 4 MB

Zprávy mohou být příliš velké pro aplikaci, frontu nebo správce front. V závislosti na prostředí produkt IBM MQ poskytuje řadu způsobů, jak se vypořádat se zprávami, které jsou delší než 4 MB.

Atribut **MaxMsgLength** můžete zvýšit na 100 MB na všech systémech IBM MQ ve verzi V6 nebo novější. Nastavte tuto hodnotu tak, aby odrážela velikost zpráv používajících frontu. Na systémech IBM MQ jiných než IBM MQ for z/OSmůžete také:

1. Použít segmentované zprávy. (Zprávy mohou být segmentovány buď aplikací, nebo správcem front.)
2. Použít referenční zprávy.

Každý z těchto přístupů je popsán ve zbytku této části.

Zvýšení maximální délky zprávy

Atribut správce front **MaxMsgLength** definuje maximální délku zprávy, kterou může správce front zpracovat. Podobně je atribut fronty **MaxMsgLength** maximální délkou zprávy, kterou lze zpracovat frontou. Výchozí maximální podporovaná délka zprávy závisí na prostředí, ve kterém pracujete.

Pokud zpracováváte velké zprávy, můžete tyto atributy změnit nezávisle na jiných platformách než z/OS. Můžete nastavit hodnotu atributu správce front v rozsahu 32768 bajtů až 100 MB.

Upozornění: V systému IBM MQ for z/OS je atribut **MaxMsgLength** správce front pevně zakódován na 100 MB.

Na všech platformách můžete nastavit hodnotu atributu fronty v rozsahu 0 až 100 MB.

Po změně jednoho nebo obou atributů **MaxMsgLength** restartujte aplikace a kanály, abyste se ujistili, že se změny projeví.

Při provádění těchto změn musí být délka zprávy menší nebo rovna atributům fronty i správce front **MaxMsgLength** . Existující zprávy však mohou být delší než jeden z atributů.

Pokud je zpráva pro frontu příliš velká, vrátí se MQRC_MSG_TOO_BIG_FOR_Q. Podobně, pokud je zpráva pro správce front příliš velká, vrátí se MQRC_MSG_TOO_BIG_FOR_Q_MGR.

Tato metoda manipulace s velkými zprávami je snadná a pohodlná. Před použitím však zvažte následující faktory:

- Uniformita mezi správci front je snížena. Maximální velikost dat zprávy je určena *MaxMsgLength* pro každou frontu (včetně přenosových front), do které bude zpráva vložena. Tato hodnota je často standardně nastavena na *MaxMsgLength*správce front, zejména pro přenosové fronty. To znesnadňuje předpovídat, zda je zpráva příliš velká, když má cestovat do vzdáleného správce front.
- Využití systémových prostředků se zvyšuje. Aplikace například potřebují větší vyrovnávací paměti a na některých platformách může dojít ke zvýšenému využití sdíleného úložiště. Úložiště fronty by mělo být ovlivněno pouze v případě, že je skutečně vyžadováno pro větší zprávy.
- Dávky kanálu jsou ovlivněny. Velká zpráva se stále počítá jako jedna zpráva k dávkovému počtu, ale potřebuje delší dobu k přenosu, čímž se zvýší doba odezvy pro ostatní zprávy.

Multi *Segmentace zpráv*

Pomocí těchto informací získáte informace o segmentaci zpráv. Tato funkce není podporována v systému IBM MQ for z/OS nebo aplikacemi používajícími IBM MQ classes for JMS.

Zvýšení maximální délky zprávy, jak je vysvětleno v tématu “Zvýšení maximální délky zprávy” na stránce 766 , má některé negativní důsledky. Může také vést k tomu, že zpráva bude příliš velká pro frontu nebo správce front. V těchto případech můžete zprávu segmentovat. Informace o segmentech viz “Skupiny zpráv” na stránce 42.

Další sekce se věnují obecnému použití pro segmentaci zpráv. Pro účely vkládání a destruktivního získávání se předpokládá, že volání MQPUT nebo MQGET *vždy* pracují v rámci pracovní jednotky. Vždy zvažte použití této techniky, abyste snížili možnost přítomnosti neúplných skupin v síti. Předpokládá se, že správce front provádí jednofázové potvrzování, ale jiné koordinační techniky jsou stejně platné.

Dále se v získávání aplikací předpokládá, že pokud více serverů zpracovává stejnou frontu, každý server spustí podobný kód, takže jeden server nikdy nenajde zprávu nebo segment, který očekává, že tam bude (protože dříve uvedl MQGMO_ALL_MSGS_AVAILABLE nebo MQGMO_ALL_SEGMENTS_AVAILABLE).

Upozornění: Při použití publikování/odběru k odesílání zpráv do tématu (nebo vkládání zpráv do aliasu tématu) není seskupení zpráv a segmentace povolena.

Vzhledem k tomu, že odběry lze vytvářet a odebírat nezávisle na aktivitě publikování, nelze zajistit, aby odběratel obdržel úplnou skupinu zpráv nebo všechny segmenty zprávy; viz RC2417: MQRC_MSG_NOT_ALLOWED_IN_GROUP.

Vložení a získání segmentované zprávy, která zahrnuje jednotky práce

Můžete vložit a získat segmentovanou zprávu, která zahrnuje jednotku práce podobným způsobem jako “Uvedení a získání skupiny, která zahrnuje jednotky práce” na stránce 757.

Do globální pracovní jednotky však nelze vkládat ani získávat segmentované zprávy.

Multi *Segmentace a opětovné sestavení podle správce front*

Jedná se o nejjednodušší scénář, kdy jedna aplikace vloží zprávu, která má být načtena jinou aplikací. Zpráva může být velká: není příliš velká pro vložení nebo aplikaci pro získávání, aby ji bylo možné zpracovat v jedné vyrovnávací paměti, ale příliš velká pro správce front nebo frontu, do které má být zpráva vložena.

Jediné změny, které jsou nezbytné pro tyto aplikace, jsou pro aplikaci provádějící vložení, aby autorizovala správce front k provedení segmentace v případě potřeby:

```
   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 aby aplikace provádějící načítání požádala 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 mohl správce front generovat jedinečný identifikátor skupiny pro každou zprávu. Pokud se tak nestane, nesouvisející zprávy mohou mít stejný identifikátor skupiny, což může následně vést k nesprávnému zpracování.

Vyrovnávací paměť aplikace musí být dostatečně velká, aby obsahovala znovu sestavovanou zprávu (pokud nezahrnete volbu MQGMO_ACCEPT_TRUNCATED_MSG).

Má-li být atribut MAXMSGLEN fronty upraven tak, aby vyhovoval segmentaci zpráv, zvažte následující:

- Minimální segment zpráv podporovaný v lokální frontě je 16 bajtů.
- Pro přenosovou frontu musí MAXMSGLEN také obsahovat prostor požadovaný pro záhlaví. Zvažte použití hodnoty alespoň o 4000 bajtů větší, než je maximální očekávaná délka uživatelských dat v jakémkoli segmentu zprávy, který by mohl být vložen do přenosové fronty.

Je-li převod dat nezbytný, aplikace provádějící získávání jej možná bude muset provést zadáním příkazu MQGMO_CONVERT. To by mělo být jednoduché, protože uživatelská procedura převodu dat je zobrazena s úplnou zprávou. Nepokoušejte se převést data v kanálu odesilatele, pokud je zpráva segmentovaná, a formát dat je takový, že uživatelská procedura převodu dat nemůže provést převod na neúplná data.

Multi *Segmentace aplikace*

Segmentace aplikací se používá v případě, že segmentace správce front není adekvátní nebo aplikace vyžadují převod dat se specifickými hranicemi segmentů.

Segmentace aplikace se používá ze dvou hlavních důvodů:

1. Samotná segmentace správce front není dostatečná, protože zpráva je příliš velká na to, aby ji mohly aplikace zpracovat v jedné vyrovnávací paměti.
2. Převod dat musí být prováděn odesílacími kanály a formát je takový, že vkládající aplikace musí stanovit, kde mají být hranice segmentu, aby byla možná konverze jednotlivého segmentu.

Pokud však převod dat nepředstavuje problém nebo pokud aplikace provádějící získávání vždy používá MQGMO_COMPLETE_MSG, lze segmentaci správce front povolit také zadáním volby 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, musí aplikace nastavit *Offset* a délku každého segmentu. V tomto případě není logický stav udržován automaticky.

Aplikace provádějící získání nemůže zaručit, že bude mít dostatečně velkou vyrovnávací paměť, aby zadržela jakoukoli znovu sestavenou zprávu. Proto musí být připraven zpracovávat segmenty jednotlivě.

Pro zprávy, které jsou segmentované, tato aplikace nechce spustit zpracování jednoho segmentu, dokud nebudou přítomny všechny segmenty, které tvoří logickou zprávu. MQGMO_ALL_SEGMENTS_AVAILABLE je proto určen pro první segment. Zadáte-li MQGMO_LOGICAL_ORDER a objeví-li se aktuální logická zpráva, bude parametr MQGMO_ALL_SEGMENTS_AVAILABLE ignorován.

Po načtení prvního segmentu logické zprávy pomocí příkazu MQGMO_LOGICAL_ORDER zajistěte, aby zbývající segmenty logické zprávy byly načteny v uvedeném pořadí.

Zprávy v různých skupinách se neberou v úvahu. Pokud se takové zprávy vyskytnou, jsou zpracovány v pořadí, ve kterém se první segment každé zprávy objeví 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 aplikací logických zpráv*

Zprávy musí být udržovány v logickém pořadí ve skupině a některé nebo všechny z nich mohou být tak velké, že vyžadují segmentaci aplikace.

V našem příkladu se má vložit skupina čtyř logických zpráv. Všechny kromě třetí zprávy jsou velké a vyžadují segmentaci, kterou provádí vkládající aplikace:

```
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 provádějící získávání je parametr MQGMO_ALL_MSGS_AVAILABLE určen v prvním příkazu MQGET. To znamená, že nebudou načteny žádné zprávy nebo segmenty skupiny, dokud nebude k dispozici celá skupina. Po načtení první fyzické zprávy skupiny se pomocí příkazu MQGMO_LOGICAL_ORDER zajistí, aby byly segmenty a zprávy skupiny načteny v uvedeném 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: Zadáte-li MQGMO_LOGICAL_ORDER a existuje-li aktuální skupina, bude parametr MQGMO_ALL_MSGS_AVAILABLE ignorován.

Referenční zprávy
Pomocí těchto informací získáte další informace o referenčních zprávách.

Poznámka: Není podporováno v produktu IBM MQ for z/OS.

Tato metoda umožňuje přenos velkého objektu z jednoho uzlu do druhého bez uložení objektu ve frontách IBM MQ ve zdrojovém nebo cílovém uzlu. To je zvláště výhodné, pokud data existují v jiné formě, například pro poštovní aplikace.

Chcete-li to provést, zadejte uživatelskou proceduru pro zprávy na obou koncích kanálu. Informace o tom, jak to provést, viz “Uživatelské programy pro zprávy kanálu” na stránce 946.

IBM MQ definuje formát záhlaví referenční zprávy (MQRMH). Popis tohoto tématu naleznete v tématu MQRMH . Tato hodnota je rozpoznána s definovaným názvem formátu a může být následována skutečnými daty.

Chcete-li zahájit přenos velkého objektu, může aplikace vložit zprávu skládající se ze záhlaví referenční zprávy bez následování dat. Protože tato zpráva opouští uzel, uživatelská procedura zprávy načte objekt odpovídajícím způsobem a připojí jej k referenční zprávě. Poté vrátí zprávu (nyní větší než dříve) odesílajícímu agentu kanálu zpráv pro přenos do přijímajícího agenta MCA.

V přijímajícím adaptéru MCA je konfigurována jiná uživatelská procedura pro zprávy. Když tato uživatelská procedura pro zprávy přijme jednu z těchto zpráv, vytvoří objekt pomocí dat objektu, která byla připojena, a předá referenční zprávu *bez* ní. Referenční zprávu může nyní přijmout aplikace a tato aplikace ví, že objekt (nebo alespoň jeho část představovaná touto referenční zprávou) byl vytvořen v tomto uzlu.

Maximální množství dat objektu, která může uživatelská procedura odeslání zprávy připojit k referenční zprávě, je omezeno vyjednanou maximální délkou zprávy pro daný kanál. Uživatelská procedura může

vrátit pouze jednu zprávu do MCA pro každou zprávu, která je předána, takže vkládající aplikace může vložit několik zpráv, aby způsobila přenos jednoho objektu. Každá zpráva musí identifikovat *logickou* délku a offset objektu, který se k ní má připojit. Avšak v případech, kdy není možné znát celkovou velikost objektu nebo maximální velikost povolenou kanálem, navrhněte odesílající uživatelskou proceduru pro odeslání zprávy tak, aby vkládající aplikace pouze vložila jedinou zprávu a uživatelská procedura sama vložila další zprávu do přenosové fronty, když připojila tolik dat, kolik mohla ke zprávě, kterou byla předána.

Před použitím této metody zpracování velkých zpráv zvažte následující body:

- Agent MCA a uživatelská procedura pro zprávy jsou spuštěny pod ID uživatele IBM MQ . Uživatelská procedura zprávy (a tedy i ID uživatele) potřebuje přístup k objektu, aby jej buď načetl na odesílajícím konci, nebo jej vytvořil na přijímajícím konci; to může být možné pouze v případech, kdy je objekt univerzálně přístupný. To vyvolává bezpečnostní problém.
- Pokud referenční zpráva s připojenými hromadnými daty musí před dosažením místa určení projít několika správci front, jsou tato hromadná data přítomna ve frontách IBM MQ v intervenujících uzlech. V těchto případech však není třeba poskytovat žádnou zvláštní podporu nebo východy.
- Návrh uživatelské procedury pro zprávy je obtížný, pokud je povoleno přesměrování nebo řazení do fronty nedoručených zpráv. V těchto případech mohou části objektu přijít mimo pořadí.
- Když do místa určení dorazí referenční zpráva, uživatelská procedura pro příjem zpráv vytvoří objekt. Toto však není synchronizováno s pracovní jednotkou agenta MCA, takže pokud je dávka vrácena zpět, další referenční zpráva obsahující stejnou část objektu dorazí v pozdější dávce a uživatelská procedura zprávy se může pokusit znovu vytvořit stejnou část objektu. Pokud je objekt například řadou aktualizací databáze, může to být nepřijatelné. Pokud ano, musí uživatelská procedura pro zprávy uchovat protokol, jehož aktualizace byly použity; to může vyžadovat použití fronty IBM MQ .
- V závislosti na charakteristice typu objektu mohou uživatelské procedury a aplikace zpráv potřebovat spolupracovat při údržbě počtu použití, aby bylo možné objekt odstranit, když již není potřeba. Identifikátor instance může být také povinný; v záhlaví referenční zprávy je pro tento účel poskytnuto pole (viz MQRMH).
- Je-li referenční zpráva vložena jako distribuční seznam, objekt musí být možné načíst pro každý výsledný distribuční seznam nebo jednotlivé místo určení v daném uzlu. Možná budete muset udržovat počty použití. Zvažte také možnost, že uzel může být konečným uzlem pro některá místa určení v seznamu, ale přechodným uzlem pro ostatní.
- Hromadná data se obvykle nepřevádějí. Důvodem je skutečnost, že k převodu dochází *před* vyvoláním uživatelské procedury pro zprávu. Z tohoto důvodu nesmí být konverze požadována na původním odesílacím kanálu. Pokud referenční zpráva prochází mezilehlým uzlem, jsou hromadná data převedena při odeslání z mezilehlého uzlu, je-li požadováno.
- Referenční zprávy nelze segmentovat.

Použití struktur MQRMH a MQMD

Popis polí v záhlaví referenční zprávy a v deskriptoru zprávy viz MQRMH a MQMD .

Ve struktuře MQMD nastavte pole *Format* na hodnotu MQFMT_REF_MSG_HEADER. Formát MQHREF, je-li požadován pro příkaz MQGET, je automaticky převeden produktem IBM MQ spolu se všemi následujícími hromadnými daty.

Zde je příklad použití polí *DataLogicalOffset* a *DataLogicalLength* modulu MQRMH:

Vkládající aplikace může vložit referenční zprávu s:

- Žádná fyzická data
- *DataLogicalLength* = 0 (tato zpráva představuje celý objekt)
- *DataLogicalOffset* = 0.

Za předpokladu, že objekt je dlouhý 70 000 bajtů, odesílající uživatelská procedura odešle prvních 40 000 bajtů podél kanálu v referenční zprávě obsahující:

- 40 000 bajtů fyzických dat po MQRMH

- *DataLogicalLength* = 40000
- *DataLogicalOffset* = 0 (od začátku objektu).

Pak umístí další zprávu do přenosové fronty obsahující:

- Žádná fyzická data
- *DataLogicalLength* = 0 (na konec objektu). Zde můžete zadat hodnotu 30 000.
- *DataLogicalOffset* = 40000 (počínaje tímto bodem).

Když je tato uživatelská procedura pro odesílání zpráv zobrazena, připojí se zbývajících 30 000 bajtů dat a pole jsou nastavena na:

- 30 000 bajtů fyzických dat po MQRMH
- *DataLogicalLength* = 30000
- *DataLogicalOffset* = 40000 (počínaje tímto bodem).

Příznak MQRMHF_LAST je také nastaven.

Popis ukázkových programů poskytovaných pro použití referenčních zpráv viz "Použití ukázkových programů na platformě Multiplatforms" na stránce 1024.

Čekání na zprávy

Chcete-li, aby program čekal na doručení zprávy do fronty, zadejte volbu MQGMO_WAIT do pole *Options* struktury MQGMO.

Použijte pole *WaitInterval* struktury MQGMO k uvedení Maximální doba (v milisekundách), po kterou má volání MQGET čekat na doručení zprávy do fronty.

Pokud zpráva v této době nedorazí, volání MQGET se dokončí s kódem příčiny MQRC_NO_MSG_AVAILABLE.

Můžete určit neomezený interval čekání pomocí konstanty MQWI_UNLIMITED v poli *WaitInterval* . Avšak události mimo vaši kontrolu mohou způsobit, že váš program bude dlouho čekat, takže tuto konstantu používejte opatrně. Aplikace IMS nesmí uvádět neomezený interval čekání, protože by to zabránilo ukončení systému IMS . (Když se IMS ukončí, vyžaduje ukončení všech závislých oblastí.) Místo toho mohou aplikace IMS určit konečný interval čekání; pokud je volání dokončeno bez načtení zprávy po tomto intervalu, zadejte další volání MQGET s volbou wait.

Poznámka: Pokud ve stejné sdílené frontě čeká více než jeden program na *odebrání* zprávy, je při příchodu zprávy aktivován pouze jeden program. Pokud však na procházení zprávy čeká více než jeden program, všechny programy lze aktivovat. Další informace viz popis pole *Options* struktury MQGMO v produktu MQGMO.

Pokud se stav fronty nebo správce front změní před vypršením intervalu čekání, dojde k následujícím akcím:

- Pokud správce front přejde do klidového stavu a použili jste volbu MQGMO_FAIL_IF_QUIESCING, bude čekání zrušeno a volání MQGET bude dokončeno s kódem příčiny MQRC_Q_MGR_QUIESCING. Bez této volby bude volání nadále čekat.
- z/OS Pokud v systému z/OSpřipojení (pro aplikaci CICS nebo IMS) přejde do klidového stavu a použili jste volbu MQGMO_FAIL_IF_QUIESCING, je čekání zrušeno a volání MQGET se dokončí s kódem příčiny MQRC_CONN_QUIESCING. Bez této volby bude volání nadále čekat.
- Je-li vynuceno zastavení správce front nebo je-li zrušeno, je volání MQGET dokončeno s kódem příčiny MQRC_Q_MGR_STOPPING nebo MQRC_CONNECTION_BROKEN.
- Pokud se změní atributy fronty (nebo fronty, na kterou se název fronty interpretuje) tak, aby požadavky na získání byly nyní zablokovány, čekání se zruší a volání MQGET se dokončí s kódem příčiny MQRC_GET_INHIBITED.
- Pokud se změní atributy fronty (nebo fronty, na kterou se název fronty interpretuje) tak, že je vyžadována volba FORCE, čekání se zruší a volání MQGET se dokončí s kódem příčiny MQRC_OBJECT_CHANGED.

z/OS Chcete-li, aby aplikace čekala ve více než jedné frontě, použijte funkci signálu IBM MQ for z/OS (viz “signalizace” na stránce 772). Další informace o okolnostech, za kterých se tyto akce vyskytují, viz MQGMO.

signalizace

Signalizace je podporována pouze na systému IBM MQ for z/OS.

Signalizace je volba ve volání MQGET, která umožňuje operačnímu systému oznamovat (nebo *signál*). program, když do fronty dorazí očekávaná zpráva. Je to jako funkce *get with wait* popsaná v tématu “Čekání na zprávy” na stránce 771 , protože umožňuje vašemu programu pokračovat v práci při čekání na signál. Pokud však používáte signalizaci, můžete uvolnit podproces aplikace a spolehnout se, že operační systém upozorní program při doručení zprávy.

Nastavení signálu

Chcete-li nastavit signál, proveďte ve struktuře MQGMO, kterou používáte ve volání MQGET, následující:

1. Nastavte volbu MQGMO_SET_SIGNAL v poli *Options* .
2. Nastavte maximální životnost signálu v poli *WaitInterval* . Tím nastavíte dobu (v milisekundách), po kterou má produkt IBM MQ monitorovat frontu. Pomocí hodnoty MQWI_UNLIMITED zadejte neomezenou životnost.

   Poznámka: Aplikace IMS nesmí uvádět neomezený interval čekání, protože by to zabránilo ukončení systému IMS . (Když se IMS ukončí, vyžaduje ukončení všech závislých oblastí.) Místo toho mohou aplikace IMS v pravidelných intervalech zkoumat stav ECB (viz krok 3). Program může mít signály nastavené na několika manipulátorů fronty současně:
3. Do pole *Signal1* zadejte adresu *bloku řízení událostí* (ECB). To vás upozorní na výsledek vašeho signálu. Úložiště ECB musí zůstat k dispozici, dokud není fronta uzavřena.

Poznámka: Volbu MQGMO_SET_SIGNAL nelze použít s volbou MQGMO_WAIT.

Když zpráva dorazí

Po doručení vhodné zprávy se ECB vrátí kód dokončení.

Kód dokončení popisuje jednu z následujících možností:

- Zpráva, pro kterou jste nastavili signál, dorazila do fronty. Zpráva není vyhrazena pro program, který požadoval signál, takže program musí znovu zadat volání MQGET, aby zprávu získal.

  Poznámka: Jiná aplikace může obdržet zprávu v době mezi přijetím signálu a zadáním dalšího volání MQGET.
- Interval čekání, který jste nastavili, vypršel a zpráva, pro kterou jste nastavili signál, nedorazila do fronty. IBM MQ zrušil signál.
- Signál byl zrušen. K tomu dochází například v případě, že se správce front zastaví 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. Pokud je chyba zjištěna okamžitě, volání se dokončí a jsou nastaveny návratové kódy.

Když je volání přijato a není okamžitě k dispozici žádná zpráva, řízení se vrátí do programu, aby mohl pokračovat v jiné práci. Není nastaveno žádné výstupní pole v deskriptoru zprávy, ale parametr **CompCode** je nastaven na hodnotu MQCC_WARNING a parametr **Reason** je nastaven na hodnotu MQRC_SIGNAL_REQUEST_ACCEPTED.

Informace o tom, co může produkt IBM MQ vrátit do aplikace při volání MQGET pomocí signalizace, naleznete v tématu MQGET.

Pokud program nemá jinou práci, kterou by měl dělat, když čeká na vyslání ECB, může čekat na ECB pomocí:

- Pro program CICS Transaction Server for z/OS příkaz EXEC CICS WAIT EXTERNAL

- V případě dávkových programů a programů IMS makro z/OS WAIT

Pokud se stav fronty nebo správce front změní, když je signál nastaven (to znamená, že ECB ještě nebyla vyslána), dojde k následujícím akcím:

- Pokud správce front přejde do klidového stavu a vy jste použili volbu MQGMO_FAIL_IF_QUIESCING, signál bude zrušen. ECB je zveřejněna s kódem dokončení MQEC_Q_MGR_QUIESCING. Bez této volby zůstane signál nastaven.
- Pokud je správce front nucen zastavit nebo je zrušen, signál je zrušen. Signál je doručen s kódem dokončení MQEC_WAIT_ZRUŠENO.
- Pokud se změní atributy fronty (nebo fronty, do které se název fronty vyřeší) tak, aby se nyní blokovaly požadavky na získání, signál se zruší. Signál je doručen s kódem dokončení MQEC_WAIT_ZRUŠENO.

Poznámka:

1. Pokud více než jeden program nastavil signál ve stejné sdílené frontě pro odebrání zprávy, pouze jeden program je aktivován příchozí zprávou. Pokud však na procházení zprávy čeká více než jeden program, všechny programy lze aktivovat. Pravidla, která správce front dodržuje při rozhodování o tom, které aplikace mají být aktivovány, jsou stejná jako pro čekající aplikace: další informace naleznete v popisu pole *Options* struktury MQGMO v části Volby MQGMO-Get-message.
2. Pokud existuje více než jedno volání MQGET čekající na stejnou zprávu s kombinací voleb čekání a signálu, je každé čekající volání považováno za stejné. Další informace viz popis pole *Options* struktury MQGMO v části Volby MQGMO-Get-message.
3. Za určitých podmínek je možné, aby volání MQGET načetlo zprávu a aby byl doručen signál (vyplývající z přijetí stejné zprávy). To znamená, že když váš program vydá další volání MQGET (protože byl signál doručen), nemůže být k dispozici žádná zpráva. Navrhněte svůj program pro testování této situace.

Informace o nastavení signálu naleznete v popisu volby MQGMO_SET_SIGNAL a v poli *Signal1* v části Signal1.

Přeskočení vrácení

Zadáním volby **MQGMO_MARK_SKIP_BACKOUT** ve volání MQGET můžete zabránit aplikačnímu programu v zadání smyčky *MQGET-error-backout* .

Poznámka: Podporováno pouze na systému IBM MQ for z/OS.

Jako součást pracovní jednotky může aplikační program zadat jedno nebo více volání MQGET pro získání zpráv z fronty. Pokud aplikační program zjistí chybu, může vrátit jednotku práce zpět. Tím se obnoví všechny prostředky aktualizované během této pracovní jednotky do stavu, ve kterém se nacházely před spuštěním pracovní jednotky, a obnoví se zprávy načtené pomocí volání MQGET.

Po obnovení jsou tyto zprávy k dispozici pro následná volání MQGET vydaná aplikačním programem. V mnoha případech to nezpůsobí problém pro aplikační program. Avšak v případech, kdy chybu vedoucí k vrácení nelze obejít, může obnovení zprávy ve frontě způsobit, že aplikační program zadá smyčku *MQGET-error-backout* .

Chcete-li se tomuto problému vyhnout, zadejte ve volání MQGET volbu MQGMO_MARK_SKIP_BACKOUT. To označí požadavek MQGET jako nezahrnutou do odvolání zahájeného aplikací; to znamená, že nesmí být odvolán. Použití této volby znamená, že když dojde k vrácení, aktualizace ostatních prostředků jsou vráceny zpět podle potřeby, ale s označenou zprávou se zachází, jako by byla načtena pod novou pracovní jednotkou.

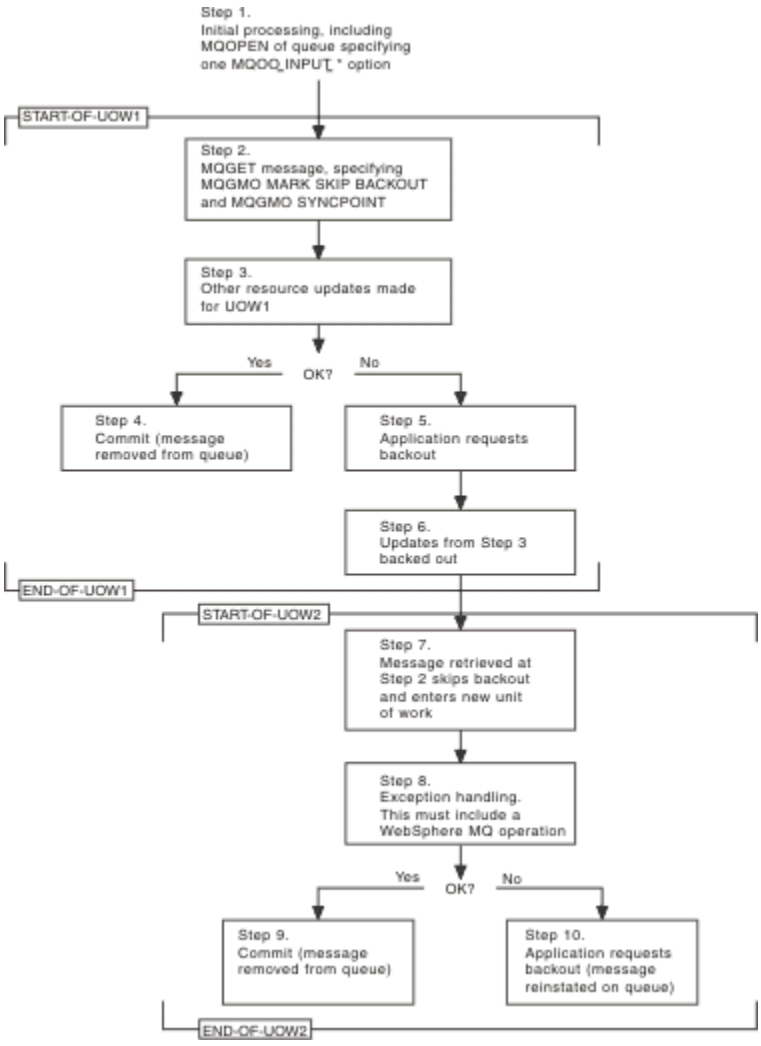
Aplikační program musí vydat volání IBM MQ buď pro potvrzení nové pracovní jednotky, nebo pro vrácení nové pracovní jednotky. Program může například provádět zpracování výjimek, například informovat původce, že zpráva byla vyřazena, a potvrdit pracovní jednotku, aby zprávu odebrali z fronty. Pokud je nová pracovní jednotka vrácena zpět (z jakéhokoli důvodu), je zpráva ve frontě obnovena.

V rámci pracovní jednotky může existovat pouze jeden požadavek MQGET označený jako přeskakující vrácení zpět; může však existovat několik dalších zpráv, které nejsou označeny jako přeskakující vrácení. Po označení zprávy jako přeskočené dojde k selhání všech dalších volání MQGET v rámci pracovní jednotky, která určila MQGMO_MARK_SKIP_BACKOUT, s kódem příčiny MQRC_SECOND_MARK_NOT_ALLOWED.

Poznámka:

1. Označená zpráva bude vynechána pouze v případě, že pracovní jednotka, která ji obsahuje, je ukončena požadavkem aplikace na její vrácení. Pokud je pracovní jednotka vrácena z jiného důvodu, zpráva je vrácena do fronty stejným způsobem, jako kdyby nebyla označena pro přeskočení.
2. Příkaz skip backout není podporován v rámci uložených procedur Db2 , které se účastní jednotek práce řízených RRS. Například volání MQGET s volbou MQGMOMARK_SKIP_BACKOUT se nezdaří s kódem příčiny MQRC_OPTION_ENVIRONMENT_ERROR.

Obrázek 63 na stránce 774 ilustruje typickou posloupnost kroků, které může aplikační program obsahovat, když je požadavek MQGET nezbytný k vynechání odvolání.



Obrázek 63. Vynechání vrácení pomocí MQGMO_MARK_SKIP_BACKOUT

Kroky v části Obrázek 63 na stránce 774 jsou následující:

Krok 1
K počátečnímu zpracování dochází v rámci transakce, včetně volání MQOPEN pro otevření fronty (zadáním jedné z voleb MQOO_INPUT_ * pro získání zpráv z fronty v kroku 2).

Krok 2
Je voláno MQGET s MQGMO_SYNCPOINT a MQGMO_MARK_SKIP_BACKOUT. MQGMO_SYNCPOINT je vyžadován, protože MQGET musí být v rámci pracovní jednotky, aby byla MQGMO_MARK_SKIP_BACKOUT efektivní. V produktu Obrázek 63 na stránce 774 se na tuto pracovní jednotku odkazuje jako na UOW1.

Krok 3
Další aktualizace prostředků se provádějí jako součást UOW1. Ty mohou zahrnovat další volání MQGET (vydané bez MQGMO_MARK_SKIP_BACKOUT).

Krok 4
Všechny aktualizace z kroků 2 a 3 jsou dokončeny podle potřeby. Aplikační program potvrdí aktualizace a UOW1 skončí. Zpráva načtená v kroku 2 je odebrána z fronty.

Krok 5
Některé aktualizace z kroků 2 a 3 nejsou dokončeny podle potřeby. Aplikační program požaduje, aby byly aktualizace provedené během těchto kroků vráceny zpět.

Krok 6
Aktualizace provedené v kroku 3 jsou vráceny zpět.

Krok 7
Požadavek MQGET provedený v kroku 2 přeskočí odvolání a stane se součástí nové pracovní jednotky UOW2.

Krok 8
UOW2 provádí zpracování výjimek v odezvě na UOW1 , které se odvolává. (Například volání MQPUT do jiné fronty, které označuje, že došlo k problému, který způsobil, že byl UOW1 odvolán.)

Krok 9
Krok 8 se dokončí podle potřeby, aplikační program potvrdí aktivitu a UOW2 skončí. Vzhledem k tomu, že požadavek MQGET je součástí UOW2 (viz krok 7), toto potvrzení způsobí odebrání zprávy z fronty.

Krok 10
Krok 8 se nedokončí podle potřeby a aplikační program provede zálohu UOW2. Protože požadavek na získání zprávy je součástí UOW2 (viz krok 7), je také odvolán a znovu zaveden do fronty. Nyní je k dispozici pro další volání MQGET vydaná tímto nebo jiným aplikačním programem (stejným způsobem jako jakákoli jiná zpráva ve frontě).

Převod dat aplikace

V případě potřeby MCA převádí deskriptor zprávy a data záhlaví na požadovanou znakovou sadu a kódování. Převod může provést buď konec spoje (tj. lokální MCA nebo vzdálený MCA).

Když aplikace vkládá zprávy do fronty, lokální správce front přidá do deskriptorů zpráv řídicí informace, aby usnadnil řízení zpráv při jejich zpracování správci front a MCA. V závislosti na prostředí jsou datová pole záhlaví zprávy vytvořena ve znakové sadě a kódování lokálního systému.

Když přesouváte zprávy mezi systémy, někdy potřebujete převést data aplikace na znakovou sadu a kódování požadované přijímajícím systémem. To lze provést buď z aplikačních programů v přijímacím systému, nebo pomocí MCA v odesílajícím systému. Pokud je konverze dat podporována v přijímajícím systému, použijte aplikační programy pro převod dat aplikace, spíše než v závislosti na převodu, který již proběhl v odesílajícím systému.

Data aplikace se převedou v rámci aplikačního programu, když zadáte volbu MQGMO_CONVERT v poli *Options* struktury MQGMO předané volání MQGET, a když *všechny* 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 ve volání MQGET.
- Pole *Format* ve struktuře MQMD přidružené ke zprávě není MQFMT_NONE.
- Hodnota *BufferLength* určená ve volání MQGET není nula.
- Délka dat zprávy není nula.
- Správce front podporuje převod mezi poli *CodedCharSetId* a *Encoding* určenými ve strukturách MQMD přidružených ke zprávě a volání MQGET. Podrobnosti o podporovaných identifikátorech kódovaných znakových sad a kódování počítače viz CodedCharSetId a Kódování .
- Správce front podporuje převod formátu zprávy. Pokud je pole *Format* struktury MQMD přidružené ke zprávě jedním z vestavěných formátů, může správce front zprávu převést. Pokud *Format* není jedním z vestavěných formátů, musíte pro převod zprávy napsat uživatelskou proceduru pro převod dat.

Má-li odesílající agent MCA převést data, zadejte klíčové slovo CONVERT (YES) v definici každého odesílacího kanálu nebo kanálu serveru, pro který je převod vyžadován. Pokud se převod dat nezdaří, je zpráva odeslána do fronty DLQ odesílajícího správce front a pole *Feedback* struktury MQDLH označuje příčinu. Pokud zprávu nelze vložit do fronty DLQ, kanál se zavře a nepřevedená zpráva zůstane v přenosové frontě. Převod dat v aplikacích spíše než při odesílání MCA se této situaci vyhýbá.

Data ve zprávě, která jsou popsána jako *znaková* data vestavěným formátem nebo uživatelskou procedurou pro převod dat, jsou zpravidla převedena z kódované znakové sady použité zprávou na požadovanou a *číselná* pole jsou převedena na požadované kódování.

Další podrobnosti o konvencích zpracování převodu používaných při převodu vestavěných formátů a informace o zápisu vlastních uživatelských procedur pro převod dat viz “Zápis uživatelských procedur převodu dat” na stránce 950. Informace o tabulkách jazykové podpory a o podporovaných kódováních počítačů viz také Národní jazyky a Strojové kódování .

Převod znaků nového řádku EBCDIC

Potřebujete-li se ujistit, že data, která odešlete z platformy EBCDIC na platformu ASCII, jsou identická s daty, která obdržíte zpět, musíte řídit převod znaků nového řádku EBCDIC.

To můžete provést pomocí přepínače závislého na platformě, který vynutí, aby produkt IBM MQ používal nezměněné převodní tabulky, ale musíte si být vědomi nekonzistentního chování, které může vést.

Problém nastává, protože znak nového řádku EBCDIC není konzistentně převeden mezi platformami nebo převodními tabulkami. V důsledku toho, pokud jsou data zobrazena na platformě ASCII, může být formátování nesprávné. To by například ztížilo vzdálenou správu systému IBM i z platformy ASCII pomocí příkazu RUNMQSC.

Další informace o převodu dat formátu EBCDIC do formátu ASCII naleznete v tématu Převod dat .

Procházení zpráv ve frontě

Pomocí těchto informací zjistíte informace o procházení zpráv ve frontě pomocí volání MQGET.

Chcete-li použít volání MQGET k procházení zpráv ve frontě, postupujte takto:

1. Pomocí volání MQOPEN otevřete frontu pro procházení a zadejte volbu MQOO_BROWSE.
2. Chcete-li procházet první zprávu ve frontě, zavolejte příkaz MQGET s volbou MQGMO_BROWSE_FIRST. Chcete-li vyhledat požadovanou zprávu, opakovaně volejte příkaz MQGET s volbou MQGMO_BROWSE_NEXT a procházejte mnoho zpráv.

   Chcete-li zobrazit všechny zprávy, musíte nastavit pole *MsgId* a *CorrelId* struktury MQMD na hodnotu null po každém volání MQGET.
3. Zavolejte MQCLOSE a zavřete frontu.

Kurzor procházení

Když otevřete frontu (MQOPEN) pro procházení, volání vytvoří kurzor procházení pro použití s voláními MQGET, která používají jednu z voleb procházení. Kurzor procházení můžete považovat za logický ukazatel, který je umístěn před první zprávou ve frontě.

Zadáním několika požadavků MQOPEN pro stejnou frontu můžete mít aktivní více než jeden kurzor procházení (z jednoho programu).

Když voláte příkaz MQGET pro procházení, použijte jednu z následujících voleb ve struktuře MQGMO:

MQGMO_BROWSE_FIRST
Získá kopii první zprávy, která splňuje podmínky určené ve struktuře MQMD.

MQGMO_BROWSE_NEXT
Získá kopii další zprávy, která splňuje podmínky určené ve struktuře MQMD.

MQGMO_BROWSE_MSG_UNDER_CURSOR
Získá kopii zprávy, na kterou aktuálně ukazuje kurzor, tj. zprávu, která byla naposledy načtena pomocí volby MQGMO_BROWSE_FIRST nebo MQGMO_BROWSE_NEXT.

Ve všech případech zpráva zůstává ve frontě.

Když otevřete frontu, kurzor procházení se umístí logicky těsně před první zprávu ve frontě. To znamená, že pokud provedete volání MQGET okamžitě po volání MQOPEN, můžete použít volbu MQGMO_BROWSE_NEXT k procházení první zprávy; nemusíte používat volbu MQGMO_BROWSE_FIRST.

Pořadí, ve kterém jsou zprávy kopírovány z fronty, je určeno atributem **MsgDeliverySequence** fronty. (Další informace viz "Pořadí, ve kterém jsou zprávy načítány z fronty" na stránce 749.)

- "Fronty v pořadí FIFO (první dovnitř, první ven)" na stránce 777
- "Fronty v posloupnosti priorit" na stránce 777
- "Nepotvrzené zprávy" na stránce 777
- "Změnit na posloupnost fronty" na stránce 777
- "Použití indexu fronty" na stránce 777

Fronty v pořadí FIFO (první dovnitř, první ven)

První zpráva ve frontě v této posloupnosti je zpráva, která byla ve frontě nejdéle.

Pomocí MQGMO_BROWSE_NEXT můžete číst zprávy sekvenčně ve frontě. Během procházení se zobrazí všechny zprávy umístěné do fronty, protože fronta v tomto pořadí obsahuje zprávy umístěné na konci. Když kurzor rozpozná, že dosáhl konce fronty, kurzor procházení zůstane tam, kde je, a vrátí se s MQRC_NO_MSG_AVAILABLE. Poté ji můžete buď nechat čekat na další zprávy, nebo ji resetovat na začátek fronty pomocí volání MQGMO_BROWSE_FIRST.

Fronty v posloupnosti priorit

První zpráva ve frontě v tomto pořadí je zpráva, která byla ve frontě nejdéle a která má nejvyšší prioritu v době vydání volání MQOPEN.

Ke čtení zpráv ve frontě použijte MQGMO_BROWSE_NEXT.

Kurzor procházení ukazuje na další zprávu, která pracuje od priority první zprávy k dokončení se zprávou s nejnižší prioritou. Prochází všechny zprávy vkládané do fronty během této doby, pokud mají prioritu rovnou nebo nižší než zpráva identifikovaná aktuálním kurzorem procházení.

Všechny zprávy vkládané do fronty s vyšší prioritou lze procházet pouze pomocí:

- Znovu se otevře fronta pro procházení. V tomto okamžiku je vytvořen nový kurzor procházení.
- Použití volby MQGMO_BROWSE_FIRST

Nepotvrzené zprávy

Nepotvrzená zpráva není nikdy viditelná pro procházení; kurzor procházení ji přeskočí.

Zprávy v rámci jednotky práce nelze procházet, dokud není jednotka práce potvrzena. Zprávy nemění svou pozici ve frontě, když jsou potvrzeny, takže vynechány, nepotvrzené zprávy nebudou zobrazeny, a to ani v případě, že *jsou* potvrzeny, pokud nepoužijete volbu MQGMO_BROWSE_FIRST a nepracujete s frontou znovu.

Změnit na posloupnost fronty

Pokud se pořadí doručení zpráv změní z priority na FIFO, zatímco jsou zprávy ve frontě, 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, která obsahuje pouze zprávy s jednou prioritou (trvalou nebo dočasnou nebo obojí), používá správce front index k procházení při použití určitých forem procházení.

Poznámka: Podporováno pouze na systému IBM MQ for z/OS.

Pokud indexovaná fronta obsahuje pouze zprávy s jednou prioritou, použije se jakákoli z následujících forem procházení:

1. Pokud je fronta indexována podle MSGID, požadavky na procházení, které předávají MSGID ve struktuře MQMD, jsou zpracovány pomocí indexu k nalezení cílové zprávy.
2. Pokud je fronta indexována pomocí CORRELID, požadavky na procházení, které předávají CORRELID ve struktuře MQMD, jsou zpracovány s použitím indexu pro vyhledání cílové zprávy.
3. Je-li fronta indexována pomocí GROUPID, požadavky na procházení, které předávají hodnotu GROUPID ve struktuře MQMD, jsou zpracovány pomocí indexu pro vyhledání cílové zprávy.

Pokud požadavek na procházení nepředá MSGID, CORRELID nebo GROUPID ve struktuře MQMD, fronta je indexována a je vrácena zpráva, musí být nalezena položka indexu pro zprávu a informace v ní použité k aktualizaci kurzoru procházení. Pokud použijete široký výběr hodnot indexu, nepřidá se k požadavku na procházení významné další zpracování.

Procházení zpráv, když 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 MQGMO_BROWSE_MSG_UNDER_CURSOR:

1. Zadejte příkaz MQGET s:
   - Buď volba MQGMO_BROWSE_FIRST, nebo MQGMO_BROWSE_NEXT
   - Volba MQGMO_ACCEPT_TRUNCATED_MSG
   - Délka vyrovnávací paměti nula

   Poznámka: Je-li pravděpodobné, že stejnou zprávu obdrží jiný program, zvažte také použití volby MQGMO_LOCK. Měla by být vrácena hodnota MQRC_TRUNCATED_MSG_ACCEPTED.
2. Použijte vrácený soubor *DataLength* k přidělení potřebného úložiště.
3. Zadejte příkaz MQGET s kurzorem MQGMO_BROWSE_MSG_UNDER_CURSOR.

Zpráva, na kterou bylo poukázáno, je poslední, která byla načtena; kurzor procházení nebude přesunut. Můžete se rozhodnout zprávu uzamknout pomocí volby MQGMO_LOCK nebo odemknout zamknutou zprávu pomocí volby MQGMO_UNLOCK.

Volání selže, pokud od otevření fronty nebyl úspěšně zadán příkaz MQGET s volbami MQGMO_BROWSE_FIRST nebo MQGMO_BROWSE_NEXT.

Odebrání zprávy, kterou jste prohlédli
Z fronty můžete odebrat zprávu, kterou jste již prohlédli, za předpokladu, že jste otevřeli frontu pro odebrání zpráv i pro procházení. (Musíte zadat jednu z voleb MQOO_INPUT_ * a také volbu MQOO_BROWSE pro volání MQOPEN.)

Chcete-li zprávu odebrat, znovu vyvolejte příkaz MQGET, ale v poli *Options* struktury MQGMO zadejte hodnotu MQGMO_MSG_UNDER_CURSOR.V tomto případě volání MQGET ignoruje pole *MsgId*, *CorrelId*a *GroupId* struktury MQMD.

V době mezi kroky procházení a odebrání může jiný program odebrat zprávy z fronty, včetně zprávy pod kurzorem procházení. V tomto případě volání MQGET vrátí kód příčiny, který říká, že zpráva není k dispozici.

Procházení zpráv v logickém pořadí

"Logické a fyzické řazení" na stránce 749 vysvětluje rozdíl mezi logickým a fyzickým pořadem 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 aplikace prochází různé zprávy jedné skupiny (pomocí logického pořadí), je důležité, aby bylo logické pořadí dodržováno, aby se dosáhlo začátku další skupiny, protože poslední zpráva jedné skupiny se může fyzicky vyskytnout *po* první zprávě další skupiny. Volba MQGMO_LOGICAL_ORDER zajišťuje, že při procházení fronty je dodržováno logické pořadí.

S péčí o operace procházení použijte volbu MQGMO_ALL_MSGS_AVAILABLE (nebo MQGMO_ALL_SEGMENTS_AVAILABLE). Zvažte případ logických zpráv s parametrem 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 nejsou, zpráva se předá. To může znamenat, že když chybějící zprávy dorazí později, nejsou si všimnuty další operací procházení.

Pokud jsou 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, vrátí se první logická zpráva skupiny 456 a ponechá se kurzor procházení na této logické zprávě. Pokud nyní dorazí 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, není si všimnuto, 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 jako celek) můžete použít MQGMO_ALL_MSGS_AVAILABLE společně s MQGMO_BROWSE_FIRST. Jinak musíte skenování procházení zopakovat, abyste si poznamenali nově příchozí zprávy, které byly vynechány; pouze vydání MQGMO_WAIT spolu s MQGMO_BROWSE_NEXT a MQGMO_ALL_MSGS_AVAILABLE je neberou v úvahu. (To se také stane se zprávami s vyšší prioritou, které mohou dorazit po dokončení skenování zpráv.)

Další sekce se zabývají příklady procházení, které se zabývají nesegmentovanými zprávami; segmentované zprávy se řídí podobnými principy.

Procházení zpráv ve skupinách
V tomto příkladu aplikace prochází každou zprávu ve frontě v logickém pořadí.

Zprávy ve frontě mohou být seskupeny. V případě seskupených zpráv aplikace nechce spustit zpracování žádné skupiny, dokud nedorazí všechny zprávy v ní obsažené. MQGMO_ALL_MSGS_AVAILABLE je proto určen pro první zprávu ve skupině; pro následné zprávy ve skupině je tato volba zbytečná.

V tomto příkladu se používá MQGMO_WAIT. Avšak i když může být čekání uspokojeno, pokud dorazí nová skupina, z důvodů uvedených v části “Procházení zpráv v logickém pořadí” na stránce 778není splněno, pokud kurzor procházení již prošel první logickou zprávou ve skupině a zbývající zprávy nyní dorazí. Nicméně čekání na vhodný interval zajistí, že aplikace nebude při čekání na nové zprávy nebo segmenty neustále cyklovat.

MQGMO_LOGICAL_ORDER se používá v rámci celého procesu, aby se zajistilo, že skenování je v logickém pořadí. To je v rozporu s destruktivním příkladem MQGET, kde vzhledem k tomu, že je každá skupina odebírána, není MQGMO_LOGICAL_ORDER použito při hledání první (nebo jediné) zprávy ve skupině.

Předpokládá se, že vyrovnávací paměť aplikace je vždy dostatečně velká, aby zadržela celou zprávu, bez ohledu na to, zda byla zpráva segmentována. Hodnota MQGMO_COMPLETE_MSG je proto určena pro každý příkaz MQGET.

Následuje 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 se nevrátí MQRC_NO_MSG_AVAILABLE.

Procházení a načítání destruktivně
V tomto příkladu aplikace prochází každou logickou zprávu ve skupině, než se rozhodne, zda tuto skupinu načíst destruktivně.

První část tohoto příkladu je podobná předchozí. Nicméně, v tomto případě, když jsme procházeli celou skupinu, jsme se rozhodli vrátit a získat ji destruktivně.

Vzhledem k tomu, že každá skupina je v tomto příkladu odebrána, není při hledání první nebo jediné zprávy ve skupině použita hodnota MQGMO_LOGICAL_ORDER.

Následuje příklad procházení a následného destruktivního načítání:

```
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 */
...
```

Vyhnutí se opakovanému doručení procházených zpráv
Pomocí určitých otevřených voleb a voleb get-message můžete označit zprávy jako procházené, aby nebyly znovu načteny aktuálními nebo jinými spolupracujícími aplikacemi. Zprávy lze explicitně nebo automaticky zrušit jejich označení, aby byly znovu k dispozici pro procházení.

Pokud procházíte zprávy ve frontě, můžete je načíst v jiném pořadí, než v jakém je načtete, pokud je máte destruktivně. Zejména můžete procházet stejnou zprávu vícekrát, což není možné, pokud je odebrána z fronty. Chcete-li se tomu vyhnout, můžete *označit* zprávy tak, jak jsou procházeny, a vyhnout se načítání označených zpráv. Někdy se nazývá *procházet se značkou*. Chcete-li označit procházené zprávy, použijte volbu získání zprávy MQGMO_MARK_BROWSE_HANDLE a načtěte pouze zprávy, které nejsou označeny, použijte MQGMO_UNMARKED_BROWSE_MSG. Pokud použijete kombinaci voleb MQGMO_BROWSE_FIRST, MQGMO_UNMARKED_BROWSE_MSG

a MQGMO_MARK_BROWSE_HANDLE a zadáte opakované MQGETs, načtete postupně každou zprávu ve frontě. To zabraňuje opakovanému doručení zpráv, i když se používá MQGMO_BROWSE_FIRST, aby se zajistilo, 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 vrácena hodnota MQRC_NO_MSG_AVAILABLE.

Pokud stejnou frontu prochází více aplikací, mohou tuto frontu otevřít pomocí voleb MQOO_CO_OP a MQOO_BROWSE. Popisovač objektu vrácený každým objektem MQOPEN je považován za součást spolupracující skupiny. Každá zpráva vrácená voláním MQGET uvádějící volbu MQGMO_MARK_BROWSE_CO_OP je považována za označenou pro tuto spolupracující sadu popisovačů.

Pokud byla zpráva po určitou dobu označena, může být automaticky odznačena správcem front a znovu zpřístupněna pro procházení. Atribut správce front MsgMarkBrowseInterval udává dobu v milisekundách, po kterou má zpráva zůstat označena pro spolupracující sadu popisovačů. Hodnota MsgMarkBrowseInterval -1 znamená, že zprávy nejsou nikdy automaticky odznačeny.

Když se zastaví jediný proces nebo sada spolupracujících procesů označujících zprávy, všechny označené zprávy se neoznačí.

Příklady kooperativního prohlížení

Chcete-li procházet zprávy ve frontě a iniciovat spotřebitele na základě obsahu jednotlivých zpráv, můžete spustit více kopií aplikace dispečera. V každém dispečeru otevřete frontu pomocí MQOO_CO_OP. To znamená, že dispečeři spolupracují a budou si vzájemně vědomi označených zpráv. Každý dispečer poté provede opakovaná volání MQGET se zadáním voleb MQGMO_BROWSE_FIRST, MQGMO_UNMARKED_BROWSE_MSG a MQGMO_MARK_BROWSE_CO_OP (tuto kombinaci voleb lze reprezentovat pomocí jediné konstanty MQGMO_BROWSE_CO_OP). Každá aplikace dispečera poté načte pouze ty zprávy, které dosud nebyly označeny jinými spolupracujícími dispečery. Dispečer inicializuje spotřebitele a předá spotřebiteli MsgToken vrácený příkazem MQGET, který destruktivně získá zprávu z fronty. Pokud spotřebitel provede zpětný příkaz MQGET zprávy, bude zpráva k dispozici pro jeden z prohlížečů k opětovnému odbavení, protože již není označena. Pokud spotřebitel neprovede operaci MQGET pro zprávu, po uplynutí MsgMarkBrowseInterval zruší správce front označení zprávy pro spolupracující sadu manipulátorů a lze ji znovu odeslat.

Namísto více kopií stejné aplikace dispečera může fronta procházet více různých aplikací dispečera, z nichž každá je vhodná pro zpracování podmnožiny zpráv ve frontě. V každém dispečeru otevřete frontu pomocí MQOO_CO_OP. To znamená, že dispečeři spolupracují a budou si vzájemně vědomi označených zpráv.

- Je-li důležité pořadí zpracování zpráv pro jeden dispečer, provede každý dispečer opakovaná volání MQGET s určením voleb MQGMO_BROWSE_FIRST, MQGMO_UNMARKED_BROWSE_MSG a MQGMO_MARK_BROWSE_HANDLE (nebo MQGMO_BROWSE_HANDLE). Pokud je procházená zpráva vhodná pro zpracování tímto dispečerem, provede volání MQGET s určením MQMO_MATCH_MSG_TOKEN, MQGMO_MARK_BROWSE_CO_OP a MsgToken vrácené předchozím voláním MQGET. Pokud je volání úspěšné, dispečer inicializuje spotřebitele a předá mu MsgToken .
- Pokud pořadí zpracování zpráv není důležité a očekává se, že dispečer zpracuje většinu zpráv, které zjistí, použijte volby MQGMO_BROWSE_FIRST, MQGMO_UNMARKED_BROWSE_MSG a MQGMO_MARK_BROWSE_CO_OP (nebo MQGMO_BROWSE_CO_OP). Pokud dispečer prochází zprávu, kterou nemůže zpracovat, zruší označení zprávy voláním MQGET s volbou MQMO_MATCH_MSG_TOKEN, MQGMO_UNMARK_BROWSE_CO_OP a dříve vráceným MsgToken .

Některé případy, kdy volání MQGET selže

Pokud jsou určité atributy fronty změněny pomocí volby FORCE v příkazu mezi vyvoláním MQOPEN a voláním MQGET, volání MQGET selže a vrátí kód příčiny MQRC_OBJECT_CHANGED.

Správce front označí popisovač objektu jako neplatný. K tomu dochází také v případě, že změny platí pro všechny fronty, pro které se název fronty interpretuje. Atributy, které tímto způsobem ovlivňují manipulátor, jsou uvedeny v popisu volání MQOPEN v části MQOPEN. Pokud volání vrátí kód příčiny MQRC_OBJECT_CHANGED, zavřete frontu, znovu ji otevřete a pokuste se znovu získat zprávu.

Pokud jsou pro frontu, ze které se pokoušíte získat zprávy (nebo libovolnou frontu, do které se název fronty interpretuje), operace get blokována, volání MQGET selže a vrátí kód příčiny MQRC_GET_INHIBITED. K tomu dochází i v případě, že pro procházení používáte volání MQGET. Pokud se později pokusíte o volání MQGET, může dojít k úspěšnému získání zprávy, pokud je návrh aplikace takový, že ostatní programy pravidelně mění atributy front.

Pokud byla odstraněna dynamická fronta (dočasná nebo trvalá), volání MQGET s použitím dříve získaného popisovače objektu se nezdaří a vrátí kód příčiny MQRC_Q_DELETED.

Zápis aplikací publikování/odběru

Začněte psát aplikace publikování/odběru IBM MQ .

Přehled konceptů publikování/odběru naleznete v tématu Zasílání zpráv publikování/odběru.

Informace o psaní různých typů aplikací publikování/odběru naleznete v následujících tématech:

- “Psaní aplikací vydavatele” na stránce 783
- “Zápis aplikací odběratele” na stránce 789
- “Životní cykly publikování/odběru” na stránce 805
- “Vlastnosti zprávy publikování/odběru” na stránce 810
- “Řazení zpráv” na stránce 811
- “Zachycování publikací” na stránce 812
- “Volby publikování” na stránce 820
- “Volby odběru” na stránce 820

Související pojmy

“Koncepty vývoje aplikací” na stránce 7
K psaní aplikací IBM MQ můžete použít výběr procedurálních nebo objektově orientovaných jazyků. Než začnete navrhovat a psát aplikace IBM MQ , seznamte se se základními koncepty produktu IBM MQ .

“Vyvíjení aplikací pro IBM MQ” na stránce 5
Můžete vyvíjet aplikace pro odesílání a příjem zpráv a pro správu správců front a souvisejících prostředků. Produkt IBM MQ podporuje aplikace napsané v mnoha různých jazycích a rámcích.

“Aspekty návrhu pro aplikace IBM MQ” na stránce 47
Když jste se rozhodli, jak mohou vaše aplikace využívat platformy a prostředí, která máte k dispozici, musíte se rozhodnout, jak používat funkce nabízené produktem IBM MQ.

“Psaní procedurální žádosti o zařazení do fronty” na stránce 699
Prostřednictvím těchto informací můžete získat informace o psaní aplikací řazení do front, o připojování a odpojování od správce front, o publikování/odběru a o otevírání a zavírání objektů.

“Psaní procedurálních aplikací klienta” na stránce 882
Co potřebujete vědět, abyste mohli psát klientské aplikace na systému IBM MQ pomocí procedurálního jazyka.

“Sestavení procedurální aplikace” na stránce 967
Aplikaci IBM MQ můžete napsat v jednom z několika procedurálních jazyků a spustit ji na několika různých platformách.

“Zpracování chyb procedurálních programů” na stránce 1004
Tyto informace vysvětlují chyby související s voláními rozhraní MQI aplikací, a to buď při volání, nebo při doručení zprávy do konečného místa určení.

Související úlohy

“Použití ukázkových procedurálních programů IBM MQ” na stránce 1023
Tyto ukázkové programy jsou napsány v procedurálních jazycích a ukazují typické použití rozhraní MQI (Message Queue Interface). Programy IBM MQ na různých platformách.

Psaní aplikací vydavatele

Začněte psát aplikace vydavatele studováním dvou příkladů. První z nich je modelován co nejpřesněji v aplikaci, která vkládá zprávy do fronty, a druhá demonstruje dynamické vytváření témat-běžnější vzor pro aplikace vydavatele.

Napsání jednoduché aplikace IBM MQ Publisher je stejně jako napsání IBM MQ aplikace typu point-to-point, která vkládá zprávy do fronty (Tabulka 121 na stránce 783). Rozdíl je v tom, že zprávy MQPUT se odesílají do tématu, nikoli do fronty.

Tabulka 121. Vzor programu IBM MQ publikování/odběru podle bodu versus publikování/odběr.

Krok	**Volání MQ z bodu na bod**	**Publikovat volání MQ**
Připojit se ke správci front	MQCONN	MQCONN
Otevřít frontu	MQOPEN	
Otevřít téma		MQOPEN
Vložit zprávu (y)	MQPUT	MQPUT
Zavřít téma		MQCLOSE
Zavřít frontu	MQCLOSE	
Odpojit od správce front	MQDISC	MQDISC

Aby to bylo konkrétní, existují dva příklady aplikací pro zveřejňování cen akcií. V prvním příkladu (“Příklad 1: Vydavatel pro pevné téma” na stránce 783), který je úzce modelován při vkládání zpráv do fronty, administrátor vytvoří definici tématu podobným způsobem jako při vytváření fronty. Programátor kóduje MQPUT pro zápis zpráv do tématu namísto jejich zápisu do fronty. Ve druhém příkladu (“Příklad 2: Vydavatel k tématu proměnné” na stránce 786) je vzor interakce programu s produktem IBM MQ podobný. Rozdíl je v tom, že programátor poskytuje téma, do kterého je zpráva zapsána, spíše než administrátor. V praxi to obvykle znamená, že řetězec tématu je obsah definovaný nebo poskytovaný jiným zdrojem, například lidským vstupem prostřednictvím prohlížeče.

Související pojmy

“Zápis aplikací odběratele” na stránce 789
Začněte psát aplikace odběratele prostudováním tří příkladů: aplikace IBM MQ , která spotřebovává zprávy z fronty, aplikace, která vytváří odběr a nevyžaduje žádnou znalost řazení do front, a nakonec příklad, který používá řazení do front i odběry.

Související odkazy

DEFINOVAT TÉMA
DISPLAYTOPIC-zobrazení
DISPLAYTPSTATUS

Příklad 1: Vydavatel pro pevné téma

Program IBM MQ pro ilustraci publikování v administrativně definovaném tématu.

Poznámka: Kompaktní styl kódování je určen pro čitelnost, nikoli pro produkční použití.

Viz výstup v části Obrázek 65 na stránce 784

```
#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 64. Jednoduchý vydavatel IBM MQ pro pevné téma.

```
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 65. Ukázka výstupu z prvního příkladu vydavatele

Následující vybrané řádky kódu ilustrují aspekty psaní aplikace vydavatele pro produkt IBM MQ.

char topicNameDefault[] = "IBMSTOCKPRICE";
V programu je definován výchozí název tématu. Můžete jej přepsat zadáním názvu jiného objektu tématu jako prvního argumentu programu.

MQCHAR resTopicStr[151];
Na `resTopicStr` ukazuje `td.ResObjectString.VSPtr` a používá jej `MQOPEN` k vrácení vyřešeného řetězce tématu. Nastavte délku `resTopicStr` o jednu větší, než je délka předaná v souboru `td.ResObjectString.VSBufSize` , abyste poskytli prostor pro ukončení s hodnotou null.

memset (resTopicStr, 0, sizeof(resTopicStr));
Inicializujte `resTopicStr` na hodnoty null, abyste se ujistili, že vyřešený řetězec tématu vrácený v `MQCHARV` má ukončenou hodnotu null.

td.ObjectType = MQOT_TOPIC
Existuje nový typ objektu pro publikování/odběr: *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, někdy nazývaný objekt administrativního tématu. V příkladu je třeba objekt tématu předem vytvořit pomocí Průzkumníka IBM MQ nebo tohoto příkazu MQSC.

```
DEFINE TOPIC(IBMSTOCKPRICE) TOPICSTR(NYSE/IBM/PRICE) REPLACE;
```

td.ResObjectString.VSPtr = resTopicStr;
Vyřešený řetězec tématu je uveden ve finálním souboru `printf` v programu. Nastavte strukturu `MQCHARV ResObjectString` pro IBM MQ , aby se vyřešený řetězec vrátil zpět do programu.

MQOPEN(Hconn, &td, MQOO_OUTPUT | MQOO_FAIL_IF_QUIESCING, &Hobj, &CompCode, &Reason);
Otevřete téma pro výstup; stejně jako otevřete frontu pro výstup.

pmo.Options = MQPMO_FAIL_IF_QUIESCING | MQPMO_RETAIN;
Chcete, aby noví odběratelé mohli obdržet publikování, a zadáním hodnoty `MQPMO_RETAIN` ve vydavateli obdrží při spuštění odběratele nejnovější publikování publikované před spuštěním odběratele jako první odpovídající publikování. Alternativou je poskytnout odběratelům publikace publikované až po spuštění odběratele. Kromě toho má odběratel možnost odmítnout příjem zachovaného publikování zadáním `MQSO_NEW_PUBLICATIONS_ONLY` ve svém odběru.

MQPUT(Hconn, Hobj, &md, &pmo, (MQLONG)strlen(publication)+1, publication, &CompCode, &Reason);
Přidejte 1 k délce řetězce předaného do `MQPUT` , aby se ukončovací znak null předal do IBM MQ jako součást vyrovnávací paměti zpráv.

Co ukazuje první příklad? Tento příklad co nejpřesněji napodobuje vyzkoušený a testovaný tradiční vzor pro psaní programů typu point to point IBM MQ . Důležitou funkcí programovacího vzoru IBM MQ je, že programátor se nezabývá odesíláním zpráv. Úlohou programátora je připojit se ke správci front a předat mu zprávy, které mají být distribuovány příjemcům. V paradigmatu dvoubodového spojení programátor otevře frontu (pravděpodobně alias frontu), kterou administrátor nakonfiguroval. Alias fronty směruje zprávy do cílové fronty, buď v lokálním správci front, nebo do vzdáleného správce front. Zatímco zprávy čekají na doručení, jsou uloženy ve frontách někde mezi zdrojem a cílem.

Ve vzoru publikování/odběru namísto otevření fronty programátor otevře téma. V našem příkladu je téma přidruženo k řetězci tématu administrátorem. Správce front předává publikování pomocí front lokálním nebo vzdáleným odběratelům, kteří mají odběry odpovídající řetězci tématu publikování. Pokud jsou publikování zachována, správce front uchová nejnovější kopii publikování, i když nyní nemá žádné odběratele. Zachované publikování je k dispozici pro postoupení budoucím odběratelům. Aplikace vydavatele nehraje žádnou roli při výběru nebo směrování publikace do místa určení; jejím úkolem je vytvářet a vkládat publikace do témat definovaných administrátorem.

Tento příklad pevného tématu je atypický pro mnoho aplikací publikování/odběru: je statický. Vyžaduje, aby administrátor definoval řetězce témat a změnil témata, která jsou publikována. Běžně musí aplikace publikování/odběru znát některý nebo celý strom témat. Možná se témata často mění, nebo i když se

témata příliš nemění, počet kombinací témat je velký a pro administrátora je příliš obtížné definovat uzel tématu pro každý řetězec tématu, na kterém může být třeba publikovat. Řetězce témat možná nejsou známy před publikováním; aplikace vydavatele může použít informace z obsahu publikování k určení řetězce tématu, nebo může mít informace o řetězcích témat, které se mají publikovat z jiného zdroje, jako např. lidský vstup z prohlížeče. Chcete-li se zaměřit na dynamičtější styly publikování, další příklad ukazuje, jak dynamicky vytvářet témata v rámci aplikace vydavatele.

Témata pár vydavatelů a odběratelů společně. Návrh pravidel nebo architektury pro pojmenování témat a jejich uspořádání ve stromech témat je důležitým krokem při vývoji řešení publikování/odběru. Pečlivě se podívejte na to, do jaké míry se organizace stromu témat váže k programům vydavatele a odběratele, a váže je k obsahu stromu témat. Zeptejte se sami sebe, zda změny ve stromu témat ovlivňují aplikace vydavatele a odběratele a jak můžete minimalizovat efekt. Vestavěná do architektury modelu publikování/odběru IBM MQ je pojem objektu administrativního tématu, který poskytuje kořenovou část nebo kořenový podstrom tématu. Objekt tématu vám dává možnost definovat kořenovou část stromu témat administrativně, což zjednodušuje programování aplikací a operace, a tím zlepšuje udržovatelnost. Pokud například implementujete více aplikací typu publikování/odběr, které mají izolované stromy témat, můžete administrativním definováním kořenové části stromu témat zaručit izolaci stromů témat, a to i v případě, že v konvencích pojmenování témat přijatých různými aplikacemi neexistuje žádná konzistence.

V praxi aplikace vydavatele pokrývají spektrum od používání pouze pevných témat, jako v tomto příkladu, a proměnných témat, jako v příštím. Produkt “Příklad 2: Vydavatel k tématu proměnné” na stránce 786 také demonstruje kombinování použití témat a řetězců témat.

Související pojmy

“Příklad 2: Vydavatel k tématu proměnné” na stránce 786
Program WebSphere MQ pro ilustraci publikování v programově definovaném tématu.

“Zápis aplikací odběratele” na stránce 789
Začněte psát aplikace odběratele prostudováním tří příkladů: aplikace IBM MQ , která spotřebovává zprávy z fronty, aplikace, která vytváří odběr a nevyžaduje žádnou znalost řazení do front, a nakonec příklad, který používá řazení do front i odběry.

Příklad 2: Vydavatel k tématu proměnné
Program WebSphere MQ pro ilustraci publikování v programově definovaném tématu.

Poznámka: Kompaktní styl kódování je určen pro čitelnost, nikoli pro produkční použití.

Viz výstup v části Obrázek 67 na stránce 787.

```
#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 66. Jednoduchý vydavatel IBM MQ pro 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 67. Ukázka výstupu z příkladu druhého vydavatele

K tomuto příkladu je třeba poznamenat několik bodů.

char topicNameDefault[] = "STOCKS";
Výchozí název tématu STOCKS definuje část řetězce tématu. Tento název tématu můžete přepsat tak, že jej zadáte jako první argument programu, nebo vyloučíte použití názvu tématu zadáním / jako prvního parametru.

char topicString[101] = "IBM/PRICE";
IBM/PRICE je výchozí řetězec tématu. Tento řetězec tématu můžete přepsat tím, že jej zadáte jako druhý argument programu.

Správce front kombinuje řetězec tématu poskytnutý objektem tématu STOCKS , "NYSE", s řetězcem tématu poskytnutým programem "IBM/PRICE" a vkládá "/" mezi dva řetězce tématu. Výsledkem je vyřešený řetězec tématu "NYSE/IBM/PRICE". Výsledný řetězec tématu je stejný jako řetězec definovaný v objektu tématu IBMSTOCKPRICE a má přesně stejný efekt.

Objekt administrativního tématu přidružený k vyřešenému řetězci tématu nemusí nutně představovat stejný objekt tématu, který vydavatel předal produktu MQOPEN . Produkt IBM MQ používá strom implicitní ve vyřešeném řetězci tématu k tomu, aby se určilo, který objekt administrativního tématu definuje atributy přidružené k publikování.

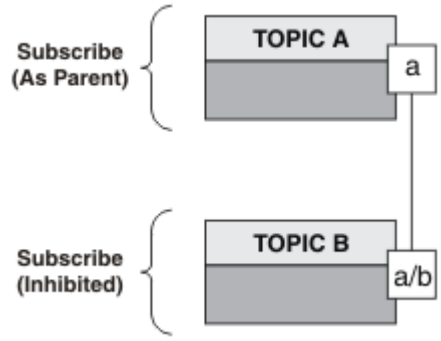
Předpokládejme, že existují dva objekty tématu A a Ba A definuje téma "a"a B definuje téma "a/b" (Obrázek 68 na stránce 788). Pokud program vydavatele odkazuje na objekt tématu A a poskytuje řetězec tématu "b", interpretace tématu na řetězec tématu "a/b", publikace zdědí své vlastnosti z objektu tématu B , protože téma odpovídá řetězci tématu "a/b" definovanému pro B.

if (strcmp(argv[1],"/"))
argv[1] je volitelně poskytovaný topicName. "/" je neplatný jako název tématu; zde označuje, že neexistuje žádný název tématu, a řetězec tématu je poskytován výhradně programem. Výstup v souboru Obrázek 67 na stránce 787 zobrazuje celý řetězec tématu, který je dynamicky dodáván programem.

strncpy(td.ObjectName, topicName, MQ_OBJECT_NAME_LENGTH);
Pro výchozí případ je třeba předem vytvořit volitelný soubor topicName pomocí Průzkumníka systému 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 68. Přidružení objektů tématu

Co ukazuje druhý příklad? I když je kód velmi podobný prvnímu příkladu-ve skutečnosti je rozdíl pouze dvou řádků-výsledkem je výrazně odlišný program od prvního. Programátor řídí cíle, do kterých jsou publikace odesílány. Ve spojení s minimálním vstupem administrátora použitým k návrhu aplikací odběratele nemusí být předdefinována žádná témata ani fronty, aby bylo možné směrovat publikování od vydavatelů k odběratelům.

V paradigmatu dvoubodového systému zpráv musí být fronty definovány dříve, než budou moci zprávy proudit. V případě publikování/odběru tak nečiní, ačkoli produkt IBM MQ implementuje publikování/odběr pomocí svého základního systému řazení do front. Výhody zaručeného doručení, transakčních transakcí a volného spojení přidruženého k systému zpráv a řazení do front jsou zděděny aplikacemi publikování/odběru.

Návrhář se musí rozhodnout, zda vydavatel a odběratel, programy mají být vědomi základního stromu témat, nebo ne, a také zda odběratel programy jsou si vědomi řazení do fronty, nebo ne. Další informace o příkladových aplikacích odběratele. Jsou navrženy pro použití s příklady vydavatele, obvykle publikování a přihlášení k odběru produktu `NYSE/IBM/PRICE`.

Související pojmy

“Příklad 1: Vydavatel pro pevné téma” na stránce 783
Program IBM MQ pro ilustraci publikování v administrativně definovaném tématu.

“Zápis aplikací odběratele” na stránce 789
Začněte psát aplikace odběratele prostudováním tří příkladů: aplikace IBM MQ , která spotřebovává zprávy z fronty, aplikace, která vytváří odběr a nevyžaduje žádnou znalost řazení do front, a nakonec příklad, který používá řazení do front i odběry.

Zápis aplikací odběratele

Začněte psát aplikace odběratele prostudováním tří příkladů: aplikace IBM MQ , která spotřebovává zprávy z fronty, aplikace, která vytváří odběr a nevyžaduje žádnou znalost řazení do front, a nakonec příklad, který používá řazení do front i odběry.

V produktu Tabulka 122 na stránce 789 jsou uvedeny tři styly odběratele nebo odběratele spolu s posloupnostmi volání funkce IBM MQ , která je charakterizují.

1. První styl, MQ Publication Consumer, je identický s programem MQ , který provádí pouze operaci MQGET. Aplikace nemá žádné znalosti o tom, že spotřebovává publikace-pouze čte zprávy z fronty. Odběr, který způsobuje směrování publikování do fronty, je vytvořen administrativně pomocí Průzkumníka IBM MQ nebo pomocí příkazu.
2. Druhý styl je upřednostňovaným vzorem pro většinu aplikací odběratele. Aplikace odběratele vytvoří odběr a poté získá publikování. Všechny správy front provádí správce front. Toto je známé jako *spravovaný odběratel*.
3. Ve třetím stylu je aplikace odběratele zodpovědná za určení fronty, která bude použita k zadržení publikování, otevření a zavření této fronty a vydání odběrů pro naplnění fronty publikacemi. Toto je známé jako *nespravovaný odběratel*.

Jedním ze způsobů, jak porozumět těmto stylům, je prostudovat příklady programů C uvedených v části Tabulka 122 na stránce 789 pro jednotlivé styly. Tyto příklady jsou navrženy tak, aby byly spuštěny ve spojení s příkladem vydavatele, který je uveden v souboru “Psaní aplikací vydavatele” na stránce 783.

Tabulka 122. Vzory programu IBM MQ pro přihlášení k odběru typu point to point vs. subscribe.

Krok	**Spotřebitel zpráv MQ**	**“Příklad 1: MQ Spotřebitel publikování” na stránce 790**	**“Příklad 2: Spravovaný odběratel MQ” na stránce 792**	**“Příklad 3: Nespravovaný odběratel produktu MQ” na stránce 797**
Připojit se ke správci front	MQCONN	MQCONN	MQCONN	MQCONN
Otevřít frontu	MQOPEN	MQOPEN		MQOPEN
Odebírat			MQSUB	MQSUB
Získat zprávu (y)	MQGET	MQGET	MQGET	MQGET
Zavřít frontu	MQCLOSE	MQCLOSE	(MQCLOSE)	MQCLOSE
Zavřít odběr			MQCLOSE	MQCLOSE

Tabulka 122. Vzory programu IBM MQ pro přihlášení k odběru typu point to point vs. subscribe. (pokračování)				
Krok	**Spotřebitel zpráv MQ**	**"Příklad 1: MQ Spotřebitel publikování" na stránce 790**	**"Příklad 2: Spravovaný odběratel MQ" na stránce 792**	**"Příklad 3: Nespravovaný odběratel produktu MQ" na stránce 797**
Odpojit od správce front	MQDISC	MQDISC	MQDISC	MQDISC

Použití MQCLOSE je vždy volitelné, buď pro uvolnění prostředků, předání voleb MQCLOSE, nebo pouze pro symetrii s MQOPEN. Vzhledem k tomu, že je nepravděpodobné, že byste museli zadat volby MQCLOSE při zavření fronty odběrů v případě odběratele Managed MQ a argument symetrie není relevantní, není fronta odběrů v Příkladu 2: Spravovaný MQ odběratelexplicitně uzavřena.

Dalším způsobem, jak porozumět vzorům aplikace publikování/odběru, je také podívat se na interakce mezi různými zúčastněnými entitami. Dobrým způsobem, jak studovat interakce, jsou diagramy posloupnosti životních linií nebo UML. Tři příklady linie jsou popsány v části "Životní cykly publikování/odběru" na stránce 805.

Příklad 1: MQ Spotřebitel publikování
Odběratel publikování MQ je spotřebitel zpráv IBM MQ , který se nepřihlašuje k odběru samotných témat.

Chcete-li vytvořit frontu odběrů a publikování pro tento příklad, spusťte následující příkazy nebo definujte objekty pomocí Průzkumníku IBM MQ .

```
DEFINE QLOCAL(STOCKTICKER) REPLACE;
DEFINE SUB(IBMSTOCKPRICESUB) DEST(STOCKTICKER) TOPICOBJ(IBMSTOCKPRICE) REPLACE;
```

Odběr IBMSTOCKPRICESUB odkazuje na objekt tématu IBMSTOCK vytvořený pro příklad vydavatele a lokální frontu STOCKTICKER. Objekt tématu IBMSTOCK definuje řetězec tématu, který se používá v odběru NYSE/IBM/PRICE. Všimněte si, že před vytvořením odběru je třeba definovat objekt tématu a frontu používanou pro příjem publikování.

Pro vzorek spotřebitele publikování produktu MQ existuje řada hodnotných fazet:

1. Multiprocessing: sdílení práce se čtením publikací. Všechna publikování přejdou do jediné fronty přidružené k tématu odběru. Frontu může otevřít více spotřebitelů pomocí MQOO_INPUT_SHARED.
2. Centrálně spravované odběry. Aplikace nevytvářejí vlastní témata odběru nebo odběry; administrátor je odpovědný za to, kam jsou publikace odesílány.
3. Koncentrace předplatného: do jedné fronty lze odeslat více různých předplatných.
4. Trvanlivost odběru: fronta přijímá všechna publikování bez ohledu na to, zda jsou spotřebitelé aktivní.
5. Migrace a koexistence: kód spotřebitele funguje stejně dobře pro dvoubodový scénář a scénář publikování/odběru.

Odběr vytvoří vztah mezi řetězcem tématu NYSE/IBM/PRICE a frontou STOCKTICKER. Publikování, včetně všech aktuálně zachovaných publikování, jsou předávány produktu STOCKTICKER od okamžiku vytvoření odběru.

Administrativně vytvořený odběr lze spravovat nebo nespravovat. Spravovaný odběr se projeví ihned po jeho vytvoření, stejně jako nespravovaný odběr. Pro spravovaný odběr nejsou k dispozici všechny fazety vzoru. Viz "Příklad 3: Nespravovaný odběratel produktu MQ" na stránce 797

Poznámka: Kompaktní styl kódování je určen pro čitelnost, nikoli pro produkční použití.

Výsledky jsou uvedeny v části Obrázek 70 na stránce 791.

```
#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 69. 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 70. Výstup ze spotřebitele publikování MQ

Existuje několik standardních rad pro programování v jazyce IBM MQ *C* , které je třeba znát:

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 hodnoty 1 do souboru `strlen(publication)`. Nastavení vyrovnávacích pamětí `MQCHAR` na hodnotu null je dobrý styl programování pro programy IBM MQ v jazyce C, které používají vyrovnávací paměti k ukládání řetězců a zajišťují, aby hodnota null následovala pole znaků, které vyrovnávací paměť zcela nenaplňuje.

MQGET(Hconn, Hobj, &md, &gmo, sizeof(publicationBuffer)-1, publication, &messlen, &CompCode, &Reason);
Vyrezervujte jednu hodnotu null na konci vyrovnávací paměti zpráv, abyste se ujistili, že vrácená zpráva má koncovou hodnotu null v případě, že `if (messlen == strlen(publication));` má hodnotu true. Tento tip doplňuje předchozí a zajišťuje, že v souboru `publicationBuffer` existuje alespoň jedna hodnota null, která není přepsána obsahem souboru `publication`.

Související pojmy

“Příklad 2: Spravovaný odběratel MQ” na stránce 792
Spravovaný odběratel produktu MQ je upřednostňovaným vzorem pro většinu aplikací odběratele. Spravovaný odběr je ten, kde produkt IBM MQ zpracovává odběr a provádí pro vás registraci a zrušení registrace. Příklad vyžaduje *ne* administrativní definici front, témat nebo odběrů.

“Příklad 3: Nespravovaný odběratel produktu MQ” na stránce 797
Nespravovaný odběratel je důležitou třídou aplikace odběratele. Díky tomu můžete kombinovat výhody publikování/odběru s *řízením* řazení do fronty a spotřeby publikací. Nespravovaný odběr je místo, kde je aplikace zodpovědná. pro určení fronty, ve které jsou uloženy odběry. Příklad demonstruje různé způsoby kombinování odběrů a front.

“Psaní aplikací vydavatele” na stránce 783
Začněte psát aplikace vydavatele studováním dvou příkladů. První z nich je modelován co nejpřesněji v aplikaci, která vkládá zprávy do fronty, a druhá demonstruje dynamické vytváření témat-běžnější vzor pro aplikace vydavatele.

Příklad 2: Spravovaný odběratel MQ
Spravovaný odběratel produktu MQ je upřednostňovaným vzorem pro většinu aplikací odběratele. Spravovaný odběr je ten, kde produkt IBM MQ zpracovává odběr a provádí pro vás registraci a zrušení registrace. Příklad vyžaduje *ne* administrativní definici front, témat nebo odběrů.

Tento nejjednodušší druh spravovaného odběratele obvykle používá *netrvalý* odběr. Příklad se zaměřuje na trvalý odběr. Odběr trvá pouze po dobu životnosti popisovače odběru z produktu `MQSUB`. Veškerá publikování, která odpovídají řetězci tématu během doby trvání odběru, jsou odesílána do fronty odběrů (a pravděpodobně i zachované publikování, pokud není příznak `MQSO_NEW_PUBLICATIONS_ONLY` nastaven nebo nastaven na výchozí hodnotu, dřívější publikování odpovídající řetězci tématu bylo zachováno a publikování bylo trvalé nebo správce front nebyl ukončen od vytvoření publikování).

S tímto vzorem můžete také použít *trvalý* odběr. Obvykle se používá spravovaný trvalý odběr, který se provádí z důvodů spolehlivosti, a nikoli k vytvoření odběru, který by bez výskytu chyb přežil odběratele. Další informace o různých životních cyklech přidružených ke spravovaným, nespravovaným, trvalým a netrvalým odběrům naleznete v sekci souvisejících témat.

Trvalé odběry jsou často přidruženy k trvalým publikacím a dočasné odběry k dočasným publikacím, ale mezi trvalostí odběru a perzistencí publikování neexistuje žádný nezbytný vztah. Všechny čtyři kombinace vytrvalosti a trvanlivosti jsou možné.

Pro uvažovaný spravovaný netrvalý případ vytvoří správce front frontu odběru, která se vymaže a odstraní při zavření fronty. Publikování jsou odebrána z fronty při zavření netrvalého odběru.

Hodnotné fazety spravovaného netrvalého vzoru, které jsou příkladem tohoto kódu, jsou následující:

1. Odběr na vyžádání: Řetězec tématu odběru je dynamický. Je poskytován aplikací při jejím spuštění.
2. Fronta s vlastní správou: fronta odběrů je definována a spravována sama.
3. Životní cyklus samosprávy odběrů: *netrvalé* odběry existují pouze po dobu trvání aplikace odběratele.

- Pokud definujete *trvalý* spravovaný odběr, bude mít za následek, že fronta trvalých odběrů a publikování budou v ní i nadále uložena, aniž by byly aktivní programy odběratele. Správce front odstraní frontu (a vymaže z ní všechna nenačtená publikování) až poté, co se aplikace nebo administrátor rozhodne odběr odstranit. Odběr lze odstranit pomocí administrativního příkazu nebo zavřením odběru s volbou MQCO_REMOVE_SUB .
- Zvažte nastavení parametru SubExpiry pro trvalé odběry tak, aby publikování přestala být odesílána do fronty a odběratel mohl před odebráním odběru spotřebovat všechna zbývající publikování a způsobit, že správce front odstraní frontu a všechna zbývající publikování v ní.

4. Flexibilní implementace řetězce tématu: Správa témat odběru je zjednodušena definováním kořenové části odběru pomocí administrativně definovaného tématu. Kořenová část stromu témat je poté v aplikaci skryta. Skrytím kořenové části lze aplikaci implementovat, aniž by aplikace neúmyslně vytvořila strom témat, který se překrývá s jiným stromem témat vytvořeným jinou instancí nebo jinou aplikací.
5. Spravovaná témata: pomocí řetězce tématu, ve kterém první část odpovídá administrativně definovanému objektu tématu, jsou publikování spravována podle atributů objektu tématu.
   - Pokud se například první část řetězce tématu shoduje s řetězcem tématu přidruženým k objektu klastrovaného tématu, může odběr přijímat publikování od jiných členů klastru.
   - Výběrové porovnávání administrativně definovaných objektů tématu a programově definovaných odběrů vám umožňuje kombinovat výhody obou. Administrátor poskytuje atributy pro témata a programátor dynamicky definuje dílčí témata bez obav o správu témat.
   - Jedná se o výsledný řetězec tématu, který se používá k porovnání objektu tématu, který poskytuje atributy přidružené k tématu, a ne nutně objekt tématu pojmenovaný v souboru sd.Objectname, ačkoli se obvykle stává jedním a stejným. Viz téma “Příklad 2: Vydavatel k tématu proměnné” na stránce 786.

Pokud v příkladu učiním odběr trvalým, budou publikování nadále odesílána do fronty odběrů poté, co odběratel uzavře odběr s volbou MQCO_KEEP_SUB . Fronta nadále přijímá publikování, když není odběratel aktivní. Toto chování můžete potlačit vytvořením odběru s volbou MQSO_PUBLICATIONS_ON_REQUEST a použitím volby MQSUBRQ pro vyžádání zachovaného publikování.

Odběr lze obnovit později otevřením odběru s volbou MQCO_RESUME .

Popisovač fronty Hobjvrácený produktem MQSUB můžete použít mnoha způsoby. Manipulátor fronty se používá v příkladu k dotazování na název fronty odběru. Spravované fronty se otevírají pomocí výchozích modelových front SYSTEM.NDURABLE.MODEL.QUEUE nebo SYSTEM.DURABLE.MODEL.QUEUE. Výchozí nastavení můžete přepsat poskytnutím vlastních trvalých a netrvalých modelových front na základě jednotlivých témat jako vlastností objektu tématu přidruženého k odběru.

Bez ohledu na atributy zděděné z modelových front nelze znovu použít manipulátor spravované fronty k vytvoření dalšího odběru. Další manipulátor pro spravovanou frontu nelze získat ani po druhém otevření spravované fronty s použitím vráceného názvu fronty. Fronta se chová, jako by byla otevřena pro výlučný 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 frontou nespravovaných odběrů.

Poznámka: Kompaktní styl kódování je určen pro čitelnost, nikoli pro produkční použití.

Výsledky jsou uvedeny v části Obrázek 73 na stránce 795.

```
#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 71. Spravovaný odběratel MQ -část 1: deklarace a obsluha parametrů.

K prohlášením v tomto příkladu je třeba učinit několik dalších poznámek.

MQHOBJ Hobj = MQHO_NONE;
Nelze explicitně otevřít frontu netrvalého spravovaného odběru pro příjem publikování, ale je třeba přidělit úložiště pro popisovač objektu, který správce front vrátí při otevření fronty. Je důležité inicializovat popisovač na hodnotu MQHO_OBJECT. Tato zpráva informuje správce front o tom, že je třeba vrátit manipulátor fronty do fronty odběrů.

MQSD sd = {MQSD_DEFAULT};
Nový deskriptor odběru použitý v souboru MQSUB.

MQCHAR48 qName;
Ačkoli tento příklad nevyžaduje znalost fronty odběrů, tento příklad se dotazuje na název fronty odběrů-vazba MQINQ je v jazyce C trochu nešikovná, takže tuto část příkladu můžete najít jako užitečnou 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 72. 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 73. Odběratel produktu MQ

K kódu v tomto příkladu je třeba přidat několik dalších komentářů.

strncpy(sd.ObjectName, topicName, MQ_Q_NAME_LENGTH);
Má-li parametr topicName hodnotu null nebo je-li prázdný *(výchozí hodnota)*, název tématu se nepoužije k výpočtu vyřešeného řetězce tématu.

sd.ObjectString.VSPtr = topicString;
Namísto použití pouze předdefinovaného objektu tématu poskytuje programátor v tomto příkladu objekt tématu a řetězec tématu, které jsou zkombinovány pomocí MQSUB. Všimněte si, že řetězec tématu je struktura 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 vyžadují příznaky sd.Options nejpečlivější pozornost. Existuje mnoho voleb, příklad uvádí pouze ty nejčastěji používané. Ostatní volby používají výchozí hodnoty.

1. Vzhledem k tomu, že odběr je *netrvalý*, tj. má životnost otevřeného odběru v aplikaci, nastavte příznak MQSO_CREATE . Můžete také nastavit příznak *(výchozí)* MQSO_NON_DURABLE pro čitelnost.
2. Doplnění 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, a to i v případě, že neexistuje žádný odběr, který by bylo možné obnovit. Selhání inicializace SubName má za následek návratový kód 2440: MQRC_SUB_NAME_ERROR z MQSUB.

   Poznámka: Parametr MQSO_RESUME je pro netrvalý spravovaný odběr vždy ignorován, ale jeho zadání bez inicializace struktury MQCHARV pro sd.SubName způsobí chybu.
3. Kromě toho existuje třetí příznak, který ovlivňuje způsob otevření odběru, MQSO_ALTER. Vzhledem ke správným oprávněním jsou vlastnosti obnoveného odběru změněny tak, aby odpovídaly jiným atributům uvedeným v souboru MQSUB.

   Poznámka: Musí být zadán alespoň jeden z příznaků MQSO_CREATE, MQSO_RESUME a MQSO_ALTER . Viz Volby (MQLONG). V souboru “Příklad 3: Nespravovaný odběratel produktu MQ” na stránce 797jsou uvedeny příklady použití všech tří příznaků.
4. Nastavte MQSO_MANAGED pro správce front tak, aby automaticky spravoval odběr.

sd.ObjectString.VSLength = MQVS_NULL_TERMINATED;
Volitelně vynechte nastavení délky MQCHARV pro ukončené řetězce s hodnotou null a místo toho použijte příznak terminátoru s hodnotou null.

sd.ResObjectString.VSPtr = resTopicStr;
Výsledný řetězec tématu je uveden v prvním souboru printf v programu. Nastavte MQCHARV ResObjectString pro IBM MQ , aby se vyřešený řetězec vrátil zpět do programu.

Poznámka: resTopicStringBuffer je inicializován na hodnoty null v memset(resTopicStr, 0, sizeof(resTopicStrBuffer)). Vrácené řetězce témat nekončí hodnotou null na konci.

sd.ResObjectString.VSBufSize = sizeof(resTopicStrBuffer)-1;
Nastavte velikost vyrovnávací paměti sd.ResObjectString na hodnotu o jednu menší, než je skutečná velikost. To zabrání přepsání ukončovače hodnoty null, který je uveden, 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 ne nutně délka vráceného řetězce. Otestujte sd.ResObjectString.VSLength < sd.ResObjectString.VSBufSiz , abyste potvrdili, že je řetězec tématu úplný.

MQSUB(Hconn, &sd, &Hobj, &Hsub, &CompCode, &Reason);
Funkce MQSUB vytvoří odběr. Pokud je netrvalý, pravděpodobně vás jeho název nezajímá, i když můžete zkontrolovat jeho stav v Průzkumníku IBM MQ . Jako vstup můžete zadat parametr sd.SubName , abyste věděli, jaký název hledat. Samozřejmě se musíte vyhnout kolizím názvů s jinými odběry.

MQCLOSE(Hconn, &Hsub, MQCO_REMOVE_SUB, &CompCode, &Reason);
Zavření odběru i fronty odběrů je volitelné. V tomto příkladu je odběr uzavřen, ale nikoli fronta. Volba MQCLOSE MQCO_REMOVE_SUB je v tomto případě i tak výchozí, protože odběr je netrvalý. Použití MQCO_KEEP_SUB je chyba.

Poznámka: *Fronta* odběru není zavřena produktem MQSUBa její popisovač Hobjzůstává platný, dokud frontu nezavře MQCLOSE nebo MQDISC. Pokud dojde k předčasnému ukončení aplikace, správce front frontu a odběr vyčistí někdy po ukončení aplikace.

Související pojmy

"Příklad 1: MQ Spotřebitel publikování" na stránce 790
Odběratel publikování MQ je spotřebitel zpráv IBM MQ , který se nepřihlašuje k odběru samotných témat.

"Příklad 3: Nespravovaný odběratel produktu MQ" na stránce 797
Nespravovaný odběratel je důležitou třídou aplikace odběratele. Díky tomu můžete kombinovat výhody publikování/odběru s *řízením* řazení do fronty a spotřeby publikací. Nespravovaný odběr je místo, kde je aplikace zodpovědná. pro určení fronty, ve které jsou uloženy odběry. Příklad demonstruje různé způsoby kombinování odběrů a front.

"Psaní aplikací vydavatele" na stránce 783
Začněte psát aplikace vydavatele studováním dvou příkladů. První z nich je modelován co nejpřesněji v aplikaci, která vkládá zprávy do fronty, a druhá demonstruje dynamické vytváření témat-běžnější vzor pro aplikace vydavatele.

Příklad 3: Nespravovaný odběratel produktu MQ
Nespravovaný odběratel je důležitou třídou aplikace odběratele. Díky tomu můžete kombinovat výhody publikování/odběru s *řízením* řazení do fronty a spotřeby publikací. Nespravovaný odběr je místo, kde je aplikace zodpovědná. pro určení fronty, ve které jsou uloženy odběry. Příklad demonstruje různé způsoby kombinování odběrů a front.

Nespravovaný vzor je častěji přidružen k *trvalým* odběrům než *netrvalým*. Obvykle je životní cyklus odběru vytvořeného nespravovaným odběratelem nezávislý na životním cyklu samotné odebírající aplikace. Pokud je odběr trvalý, přijímá publikování i v případě, že není aktivní žádná odebírající aplikace.

Chcete-li dosáhnout stejného výsledku, můžete vytvořit trvalé *spravované* odběry, ale některé aplikace vyžadují větší flexibilitu a kontrolu nad frontami a zprávami, než je možné u spravovaného odběru. V případě trvalého spravovaného odběru vytvoří správce front trvalou frontu pro publikování, která odpovídají tématu odběru. Při odstranění odběru odstraní frontu a přidružená publikování.

Trvalé *spravované* odběry se obvykle používají v případě, že životní cyklus aplikace a odběru je v podstatě stejný, ale těžko zaručitelný. Po vytvoření trvalého odběru a po vytvoření trvalých publikování vydavatelem se v případě předčasného ukončení správce front nebo odběratele neobjeví žádné ztracené zprávy, které by bylo nutné obnovit.

V případě aplikací jiných než JMS nebo aplikací JMS, které nepoužívají sdílený odběr, správce front implicitně otevře frontu trvalého spravovaného odběru pro odběratele tak, aby nebylo možné sdílené zpracování fronty. Kromě toho, pokud vaše aplikace nepoužívá sdílené odběry JMS, není možné vytvořit více než jeden odběr pro každou spravovanou frontu a může se stát, že fronty budou obtížněji spravovány, protože máte menší kontrolu nad názvy front. Z těchto důvodů zvažte, zda je odběratel *unmanaged* MQ vhodnější pro aplikace vyžadující trvalé odběry než odběratel *managed* MQ .

Kód v souboru Obrázek 76 na stránce 802 demonstruje vzor nespravovaného trvalého odběru. Pro ilustraci kód také vytváří nespravované, dočasné odběry. Tento příklad ilustruje následující fazety vzoru:

- Odběry na vyžádání: řetězce témat odběru jsou dynamické. Jsou poskytovány aplikací při jejím spuštění.
- Zjednodušená správa témat odběru: Správa témat odběru je zjednodušena definováním kořenové části řetězce tématu odběru pomocí administrativně definovaného tématu. Tím se skryje kořenová část stromu témat z aplikace. Skrytím kořenové části může být odběratel implementován do různých stromů témat.
- Flexibilní správa předplatného: předplatné můžete definovat buď administrativně, nebo jej vytvořit na vyžádání v programu odběratele. Neexistuje žádný rozdíl mezi administrativně a programově vytvořenými odběry, kromě atributu, který ukazuje, jak byl odběr vytvořen. Existuje třetí typ odběru,

který je automaticky vytvořen správcem front pro distribuci odběrů. Všechny odběry jsou zobrazeny v Průzkumníku IBM MQ .

- Flexibilní přidružení odběrů k frontám: Předdefinovaná lokální fronta je přidružena k odběru funkcí MQSUB . Existují různé způsoby použití MQSUB pro přidružení odběrů k frontám:
  - Přidružte odběr k frontě, která *nemá* existující odběry `MQSO_CREATE + (Hobj from MQOPEN)`.
  - Přidružte *nový* odběr k 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 zajistit různé vstupní stavy odběru a fronty, aniž byste museli kódovat více verzí produktu MQSUB s různými hodnotami `sd.Options` .
  - Případně pomocí kódu specifické volby `MQSO_CREATE | MQSO_RESUME | MQSO_ALTER` správce front vrátí chybu (Tabulka 123 na stránce 799). pokud jsou stavy odběru a fronty poskytnuté jako vstup pro produkt MQSUB nekonzistentní s hodnotou parametru `sd.Options`. Obrázek 82 na stránce 805 zobrazuje výsledky zadání MQSUB pro odběr X s různými individuálními nastaveními příznaku `sd.Options` a předání tří různých popisovačů objektů.

  Prozkoumejte různé vstupy do ukázkového programu v produktu Obrázek 75 na stránce 801 , abyste se seznámili s těmito různými druhy chyb. Jednou z běžných chyb, `RC = 2440`, která není zahrnuta v případech uvedených v tabulce, je chyba názvu odběru. je obvykle způsoben předáním nulového nebo neplatného názvu odběru s `MQSO_RESUME` nebo `MQSO_ALTER`.
- Multiprocessing: Můžete sdílet mezi mnoha spotřebiteli práci čtení publikací. Všechna publikování přejdou do jediné fronty přidružené k tématu odběru. Spotřebitelé mají možnost otevřít frontu přímo pomocí produktu `MQOPEN` nebo obnovit odběr pomocí produktu `MQSUB`.
- Koncentrace odběrů: ve stejné frontě lze vytvořit více odběrů. S touto funkcí buďte opatrní, protože může vést k překrývání odběrů a k několikanásobnému příjmu stejného publikování. Volba `MQSO_GROUP_SUB` eliminuje duplicitní publikování způsobená překrývajícími se odběry.
- Odběratel a oddělení spotřebitelů: Kromě tří modelů spotřebitelů uvedených v příkladech je dalším modelem oddělení spotřebitele od odběratele. Jedná se o variantu nespravovaného odběratele produktu MQ , ale namísto vydání `MQOPEN` a `MQSUB` ve stejném programu jeden program odebírá publikace a jiný program je spotřebovává. Odběratel může být například součástí klastru publikování/odběru a odběratel připojený ke správci front mimo klastr správců front. Odběratel přijímá publikování prostřednictvím standardního distribuovaného řazení do front definováním fronty odběrů jako definice vzdálené fronty.

Porozumění chování produktu `MQSO_CREATE | MQSO_RESUME | MQSO_ALTER` je důležité, zejména pokud plánujete zjednodušit kód pomocí kombinací těchto voleb. Prohlédněte si tabulku Tabulka 123 na stránce 799 , která zobrazuje výsledky předání různých manipulátorů front do MQSUB, a výsledky spuštění ukázkového programu zobrazeného v Obrázek 77 na stránce 803 do Obrázek 82 na stránce 805.

Scénář použitý k vytvoření tabulky má jeden odběr X a dvě fronty A a B. Parametr názvu odběru `sd.SubName` je nastaven na hodnotu X, což je název odběru připojeného k frontě A. K frontě B není připojen žádný odběr.

V produktu Tabulka 123 na stránce 799je předána položka MQSUB odběr X a popisovač fronty do fronty A. Výsledky voleb odběru jsou následující:

- `MQSO_CREATE` selže, protože popisovač fronty odpovídá frontě A , která již má odběr pro X. Porovnejte toto chování s úspěšným voláním. Toto volání je úspěšné, protože k frontě B není připojen odběr produktu X .
- Produkt `MQSO_RESUME` uspěje, protože popisovač fronty odpovídá frontě A , která již má odběr produktu X. Naproti tomu volání selže v případě, že odběr X ve frontě neexistuje A.

- Produkt MQSO_ALTER se chová podobně jako produkt MQSO_RESUME s ohledem na otevření odběru a fronty. Avšak pokud se atributy obsažené v deskriptoru odběru předané do MQSUB liší od atributů odběru, MQSO_RESUME selže, zatímco MQSO_ALTER uspěje, pokud má instance programu oprávnění měnit atributy. Všimněte si, že řetězec tématu v odběru nikdy nemůžete změnit; namísto vrácení chyby produkt MQSUB ignoruje hodnoty názvu tématu a řetězce tématu v deskriptoru odběru a použije hodnoty v existujícím odběru.

Dále se podívejte na téma Tabulka 123 na stránce 799 , kde MQSUB předává odběr X a manipulátor 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 selhává. Produkt MQSUB vyhledá odběr X ve frontě B a nenalezne jej, ale namísto vrácení hodnoty *RC = 2428-odběr X neexistuje*vrátí hodnotu *RC = 2019-fronta odběru neodpovídá manipulátoru objektu fronty*. Chování třetí volby MQSO_ALTER naznačuje příčinu této neočekávané chyby. Produkt MQSUB očekává, že popisovač fronty bude ukazovat na frontu s odběrem. Nejprve zkontroluje, zda odběr uvedený v souboru sd.SubName existuje.
- Produkt MQSO_ALTER uspěje a přesune odběr z fronty A do fronty B.

Případ, který není uveden v tabulce, je případ, kdy se název odběru ve frontě A neshoduje s názvem odběru v souboru sd.SubName. Toto volání selže s *RC = 2428-odběr X neexistuje ve frontě A*.

Tabulka 123. Chyby MQSUB s různými manipulátory front a kombinacemi odběrů

Manipulátory front	**Fronta A** **Odběr X** **Fronta B** **Bez předplatného**	**Fronta A** **Bez předplatného** **Fronta B** **Bez předplatného**
Hobj pro *frontu A* předaný MQSUB	**MQSO_CREATE** RC = 2432-Odběr X již ve frontě A existuje **MQSO_RESUME** Obnoví odběr X ve frontě A **MQSO_ALTER** Pokračuje v odběru X ve frontě A a provádí povolené změny.	**MQSO_CREATE** Vytvoří odběr X ve frontě A **MQSO_RESUME** RC = 2428-Odběr X neexistuje ve frontě A **MQSO_ALTER** RC = 2428-Odběr X neexistuje ve frontě A
Objekt Hobj pro *Fronta B* byl předán objektu MQSUB.	**MQSO_CREATE** Vytvoří nový odběr X ve frontě B **MQSO_RESUME** RC = 2019-Fronta odběru neodpovídá manipulátoru 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 ve frontě B neexistuje **MQSO_ALTER** RC = 2428-odběr X ve frontě B neexistuje

Tabulka 123. Chyby MQSUB s různými manipulátory front a kombinacemi odběrů (pokračování)		
Manipulátory front	**Fronta A** **Odběr X** **Fronta B** **Bez předplatného**	**Fronta A** **Bez předplatného** **Fronta B** **Bez předplatného**
MQHO_NONE předáno MQSUB	**MQSO_CREATE** RC = 2019-Chybný popisovač objektu: nastavte příznak MQSO_MANAGED pro vytvoření spravovaného odběru a vytvoření spravované fronty **MQSO_RESUME** Pokračuje v odběru X ve frontě A a vrací Hobj do fronty A **MQSO_ALTER** Obnoví odběr X ve frontě A, vrátí Hobj do fronty A a provede povolené změny.	**MQSO_CREATE** RC = 2019-Chybný popisovač objektu: nastavte příznak MQSO_MANAGED pro vytvoření spravovaného odběru a vytvoření spravované fronty **MQSO_RESUME** RC = 2428-Žádné předplatné X **MQSO_ALTER** RC = 2019-Chybný popisovač objektu: Žádná fronta A nebo B

Poznámka: Kompaktní styl kódování je určen pro čitelnost, nikoli pro produkční použití.

```
#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 74. 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 75. Nespravovaný odběratel produktu MQ -část 2: zpracování parametrů.

Další komentáře týkající se obsluhy parametrů v tomto příkladu jsou následující:

switch((argv[5][0]))
Máte možnost zadat A lter | C reate | R esume v parametru 5, abyste otestovali účinek přepsání části nastavení volby MQSUB , které se standardně používá v příkladu. 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 nastaveno a odkazovat se na odběr, který lze obnovit nebo změnit.

***subscriptionQueue = '\0';**
sdOptions = sdOptions + MQSO_MANAGED;
Pokud je výchozí fronta odběrů STOCKTICKER nahrazena řetězcem s hodnotou null, pokud je nastaven parametr MQSO_CREATE , příklad nastaví příznak MQSO_MANAGED a vytvoří dynamickou frontu odběrů. Pokud je v pátém parametru nastavena hodnota Alter or Resume , bude chování příkladu záviset na hodnotě subscriptionName.

```
*subscriptionName = '\0';
sdOptions = sdOptions - MQSO_DURABLE;
```

Pokud je výchozí odběr IBMSTOCKPRICESUBnahrazen řetězcem s hodnotou null, příklad odebere příznak MQSO_DURABLE . Spustíte-li příklad s výchozími hodnotami pro ostatní parametry, vytvoří se další dočasný odběr určený pro produkt STOCKTICKER a obdrží duplicitní publikování. Při příštím spuštění 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 76. Nespravovaný odběratel produktu MQ -část 3: tělo kódu.

Další komentáře ke kódu v tomto příkladu jsou následující:

if (strlen(subscriptionQueue))
Pokud neexistuje žádný název fronty odběrů, příklad použije jako hodnotu parametru Hobjhodnotu MQHO_NONE .

MQOPEN(...);
Fronta odběrů se otevře a manipulátor fronty se uloží do souboru Hobj.

MQSUB(Hconn, &sd, &Hobj, &Hsub, &CompCode, &Reason);
Odběr se otevře s použitím položky Hobj předané z adresáře MQOPEN (nebo MQHO_NONE , pokud neexistuje žádný název fronty odběrů). Nespravovanou frontu lze obnovit bez explicitního otevření pomocí 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 MQCO_KEEP_SUB nebo MQCO_REMOVE_SUB. Trvalý odběr můžete zavřít pomocí produktu MQCO_REMOVE_SUB, ale nemůžete zavřít netrvalý odběr pomocí produktu MQCO_KEEP_SUB. Akce produktu MQCO_REMOVE_SUB spočívá v odebrání odběru, který zastaví veškerá další publikování odesílaná do fronty odběrů.

MQCLOSE(Hconn, &Hobj, MQCO_NONE, &CompCode, &Reason);
Není-li odběr nespravovaný, neprovede se žádná speciální akce. Pokud je fronta spravována a odběr je uzavřen s explicitním nebo implicitním produktem MQCO_REMOVE_SUB, budou všechna publikování z fronty a fronty v tomto bodě odstraněna.

gmo.MatchOptions = MQMO_MATCH_CORREL_ID;
memcpy(md.CorrelId, sd.SubCorrelId, MQ_CORREL_ID_LENGTH);
Ujistěte se, že přijaté zprávy jsou ty pro naše předplatné.

Výsledky z příkladu ilustrují aspekty publikování/odběru:

V souboru Obrázek 77 na stránce 803 začíná příklad publikováním souboru 130 v tématu 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 77. Publikovat 130 na NYSE/IBM/PRICE

Při Obrázek 78 na stránce 803 provedení příkladu s výchozími parametry obdrží zachované publikování 130. Poskytnutý objekt tématu a řetězec tématu jsou ignorovány, jak ukazuje Obrázek 82 na stránce 805. Objekt tématu a řetězec tématu jsou vždy převzaty z objektu odběru, pokud je zadán, a řetězec tématu je neměnný. Skutečné chování příkladu závisí na volbě nebo kombinaci MQSO_CREATE, MQSO_RESUMEa MQSO_ALTER. V tomto příkladu 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 78. Přijmout zachované publikování

V (Obrázek 79 na stránce 804) nejsou přijata žádná publikování, protože trvalý odběr již zachované publikování obdržel. V tomto příkladu je odběr obnoven zadáním pouze názvu odběru bez názvu fronty. Pokud byl zadán název fronty, fronta se nejprve otevře a popisovač se předá do adresáře MQSUB.

Poznámka: Chyba 2038 z MQINQ je způsobena implicitním MQOPEN STOCKTICKER od MQSUB nezahrnujících volbu MQOO_INQUIRE . Vyhněte se návratovému kódu 2038 z MQINQ explicitním otevřením fronty.

```
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 79. Pokračovat v odběru

V produktu Obrázek 80 na stránce 804příklad vytvoří netrvalý nespravovaný odběr s použitím parametru STOCKTICKER jako místa určení. Protože 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 80. Přijmout zachované publikování s novým nespravovaným netrvalým odběrem

Chcete-li v produktu Obrázek 81 na stránce 804demonstrovat překrývající se odběry, odešle se další publikace, čímž se zachované publikování změní. Dále je vytvořen nový netrvalý a nespravovaný odběr bez zadání názvu odběru. Zachované publikování je přijato dvakrát, jednou pro nový odběr a jednou pro trvalý odběr produktu IBMSTOCKPRICESUB , který je stále aktivní ve frontě STOCKTICKER . Příklad je ilustrací toho, že se jedná o frontu, která má odběry, a nikoli aplikaci. Přestože se v tomto vyvolání aplikace neodkazuje na odběr IBMSTOCKPRICESUB , aplikace obdrží publikaci dvakrát: jednou z trvalého odběru, který byl vytvořen administrativně, a jednou z netrvalého odběru vytvořeného samotnou aplikací.

```
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 81. Překrývající se odběry

V souboru Obrázek 82 na stránce 805 příklad demonstruje, že poskytnutí nového řetězce tématu a existujícího odběru nepovede ke změně odběru.

1. V prvním případě produkt `Resume` obnoví existující odběr, jak můžete očekávat, a ignoruje změněný řetězec tématu.
2. Ve druhém případě `Alter` způsobí chybu `RC = 2510, Topic not alterable`.
3. Ve třetím příkladu `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 82. Témata odběru nelze změnit.

Související pojmy

“Příklad 1: MQ Spotřebitel publikování” na stránce 790
Odběratel publikování MQ je spotřebitel zpráv IBM MQ , který se nepřihlašuje k odběru samotných témat.

“Příklad 2: Spravovaný odběratel MQ” na stránce 792
Spravovaný odběratel produktu MQ je upřednostňovaným vzorem pro většinu aplikací odběratele. Spravovaný odběr je ten, kde produkt IBM MQ zpracovává odběr a provádí pro vás registraci a zrušení registrace. Příklad vyžaduje *ne* administrativní definici front, témat nebo odběrů.

“Psaní aplikací vydavatele” na stránce 783
Začněte psát aplikace vydavatele studováním dvou příkladů. První z nich je modelován co nejpřesněji v aplikaci, která vkládá zprávy do fronty, a druhá demonstruje dynamické vytváření témat-běžnější vzor pro aplikace vydavatele.

Životní cykly publikování/odběru

Při navrhování aplikací pro publikování/odběr zvažte životní cykly témat, odběrů, odběratelů, publikací, vydavatelů a front.

Životní cyklus objektu, například odběr, začíná jeho vytvořením a končí jeho odstraněním. Může také zahrnovat další stavy a změny, které prochází, jako např. dočasné pozastavení, nadřízená a podřízená témata, vypršení platnosti a odstranění.

Tradičně jsou objekty IBM MQ , jako jsou fronty, vytvářeny administrativně nebo administrativními programy pomocí programovatelného formátu příkazů (PCF). Publikování/odběr se liší v poskytování příkazových slov rozhraní API MQSUB a MQCLOSE pro vytváření a odstraňování odběrů s koncepcí spravovaných odběrů, které nejen vytvářejí a odstraňují fronty, ale také vyčistí nespotřebované zprávy a mají přidružení mezi administrativně vytvořenými objekty tématu a programově nebo administrativně vytvořenými řetězci tématu.

Tato funkční bohatost zajišťuje širokou škálu požadavků na publikování/odběr a také zjednodušuje návrh některých běžných vzorů aplikace publikování/odběru. Spravované odběry například zjednodušují programování i administraci odběru, který má trvat pouze po dobu trvání programu, který jej vytvořil. Nespravované odběry zjednodušují programování tam, kde existuje volnější připojení mezi odběrem a odběrem publikací. Centrálně vytvořená předplatná jsou užitečná v případech, kdy je vzor jedním ze směrování publikačního provozu pro spotřebitele na základě centralizovaného modelu řízení, například odesílání informací o letu do automatizovaných bran, zatímco programově vytvořená předplatná mohou být použita v případě, že jsou zaměstnanci bran odpovědní za přihlášení k odběru záznamů cestujících pro daný let, a to zadáním čísla letu u brány.

V tomto posledním příkladu může být vhodný spravovaný trvalý odběr: Spravovaný, protože odběry jsou vytvářeny velmi často a mají jasný koncový bod, když se brána zavře a odběr může být programově odebrán; trvalý, aby se zabránilo ztrátě záznamu cestujícího kvůli tomu, že se program odběratele brány z nějakého důvodu vypne.[8]. Chcete-li zahájit zveřejňování záznamů cestujících k bráně, možný návrh by byl pro aplikaci brány, aby se oba přihlásili k záznamům cestujících pomocí čísla brány a zveřejnili otevření brány pomocí čísla brány. Vydavatel reaguje na událost otevření brány zveřejněním záznamů cestujících-které pak mohou jít i k dalším zájemcům, jako je fakturace, zaznamenávat let probíhá, a na služby zákazníkům, na textové oznámení na mobilní telefony cestujících s číslem brány.

[8] Vydavatel musí odesílat záznamy cestujících jako trvalé zprávy, aby se zabránilo dalším možným selháním, samozřejmě.

Centrálně spravovaný odběr může používat trvalý nespravovaný model, který pomocí předdefinované fronty pro každou bránu směruje seznamy cestujících k bráně.

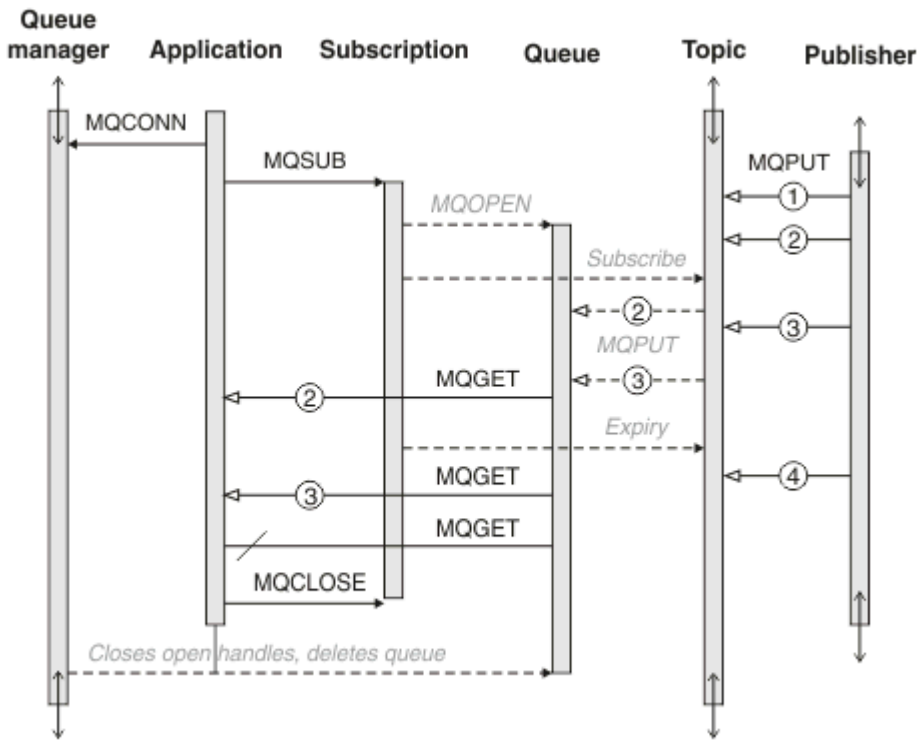
Následující tři příklady životních cyklů publikování/odběru ilustrují, jak spravovaní netrvalí, spravovaní trvalí a nespravovaní trvalí odběratelé interaktivně spolupracují s odběry, tématy, frontami, vydavateli a správcem front a jak mohou být odpovědnosti rozděleny mezi administrační programy a programy odběratele.

Spravovaný netrvalý odběratel

Obrázek 83 na stránce 806 zobrazuje aplikaci, která vytváří spravovaný netrvalý odběr, získává dvě zprávy, které jsou publikovány v tématu identifikovaném v odběru, a ukončuje se. Interakce označené šedým kurzívou s tečkovanými šipkami jsou implicitní.

Tam jsou některé body na vědomí.

1. Aplikace vytvoří odběr pro téma, které již bylo dvakrát publikováno. Když odběratel obdrží své první publikování, obdrží *druhé* publikování, které je aktuálně zachováno.
2. Správce front vytvoří dočasnou frontu odběrů a vytvoří odběr pro dané téma.
3. Odběr má vypršení platnosti. Dojde-li k vypršení platnosti odběru, nebudou do tohoto odběru odesílána žádná další publikování týkající se daného tématu, odběratel však bude i nadále získávat zprávy publikované před vypršením platnosti odběru. Vypršení platnosti publikování není ovlivněno vypršením platnosti odběru.
4. Čtvrté publikování není umístěno do fronty odběrů, a proto poslední MQGET nevrací publikování.
5. Odběratel sice svůj odběr zavře, ale nezavře připojení k frontě ani ke správci front.
6. Správce front se vyčistí krátce po ukončení aplikace. Vzhledem k tomu, že odběr je spravovaný a netrvalý, je fronta odběrů odstraněna.



Obrázek 83. Linie spravovaných netrvalých odběratelů

Spravovaný trvalý odběratel

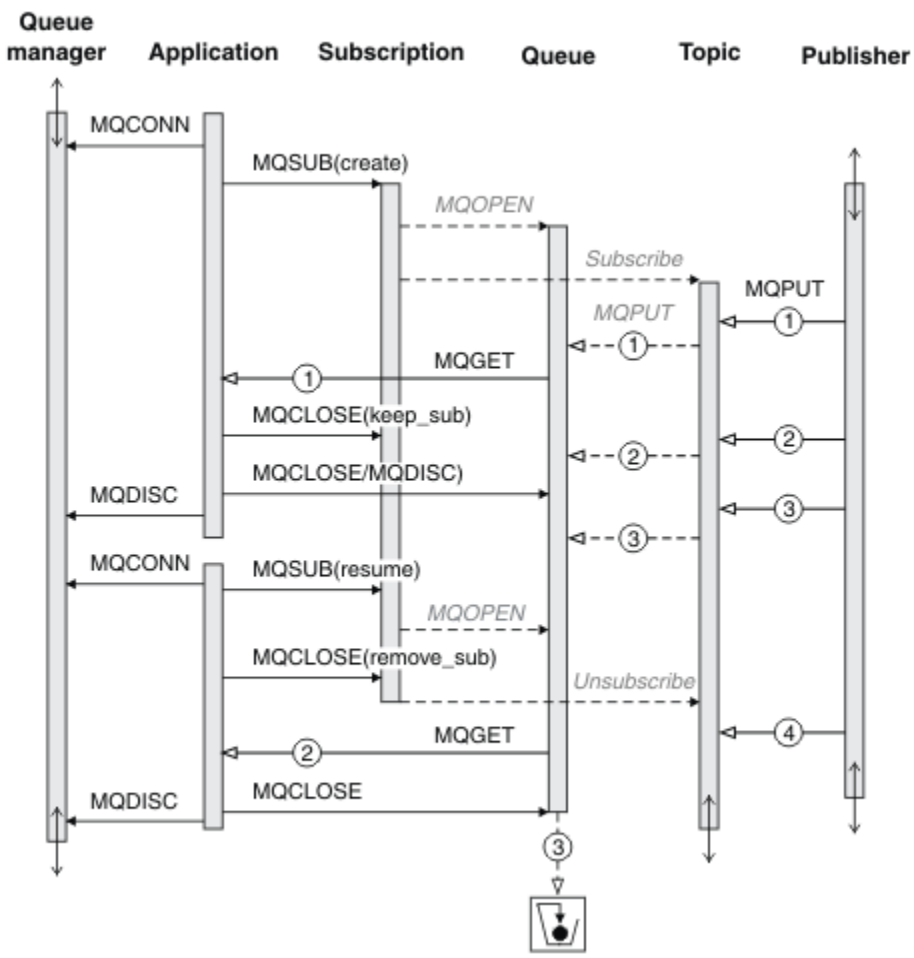
Spravovaný trvalý odběratel posouvá předchozí příklad o krok dále a zobrazuje spravovaný odběr, který přežil ukončení a restartování odebírající aplikace.

Tam jsou některé nové body na vědomí.

1. V tomto příkladu, na rozdíl od posledního, téma publikování neexistovalo před tím, než bylo definováno v odběru.
2. Při prvním ukončení odběratele dojde k zavření odběru s volbou MQCO_KEEP_SUB. Jedná se o výchozí chování pro implicitní zavření spravovaného trvalého odběru.
3. Když odběratel obnoví odběr, fronta odběrů se znovu otevře.
4. Nové publikování 2umístěné do fronty před opětným otevřením je pro produkt MQGETk dispozici i po odebrání odběru.

   I když je odběr trvalý, odběratel spolehlivě obdrží všechny zprávy odeslané vydavatelem pouze v případě, že *oba* odběr je trvalý a zprávy jsou trvalé. Perzistence zpráv závisí na nastavení pole Persistent v souboru MQMD zprávy odeslané vydavatelem. Odběratel nad tímto nemá žádnou kontrolu.
5. Zavřením odběru s příznakem MQCO_REMOVE_SUB odeberete odběr a zastavíte všechna další publikování zařazená do fronty odběrů. Po zavření fronty odběrů správce front odebere nepřečtené publikování 3a poté frontu 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 IBM MQ . Správce front nemůže spolehlivě spravovat odběr, pokud nemá úplnou kontrolu.



Obrázek 84. Linie spravovaných trvalých odběratelů

Nespravovaný trvalý odběratel

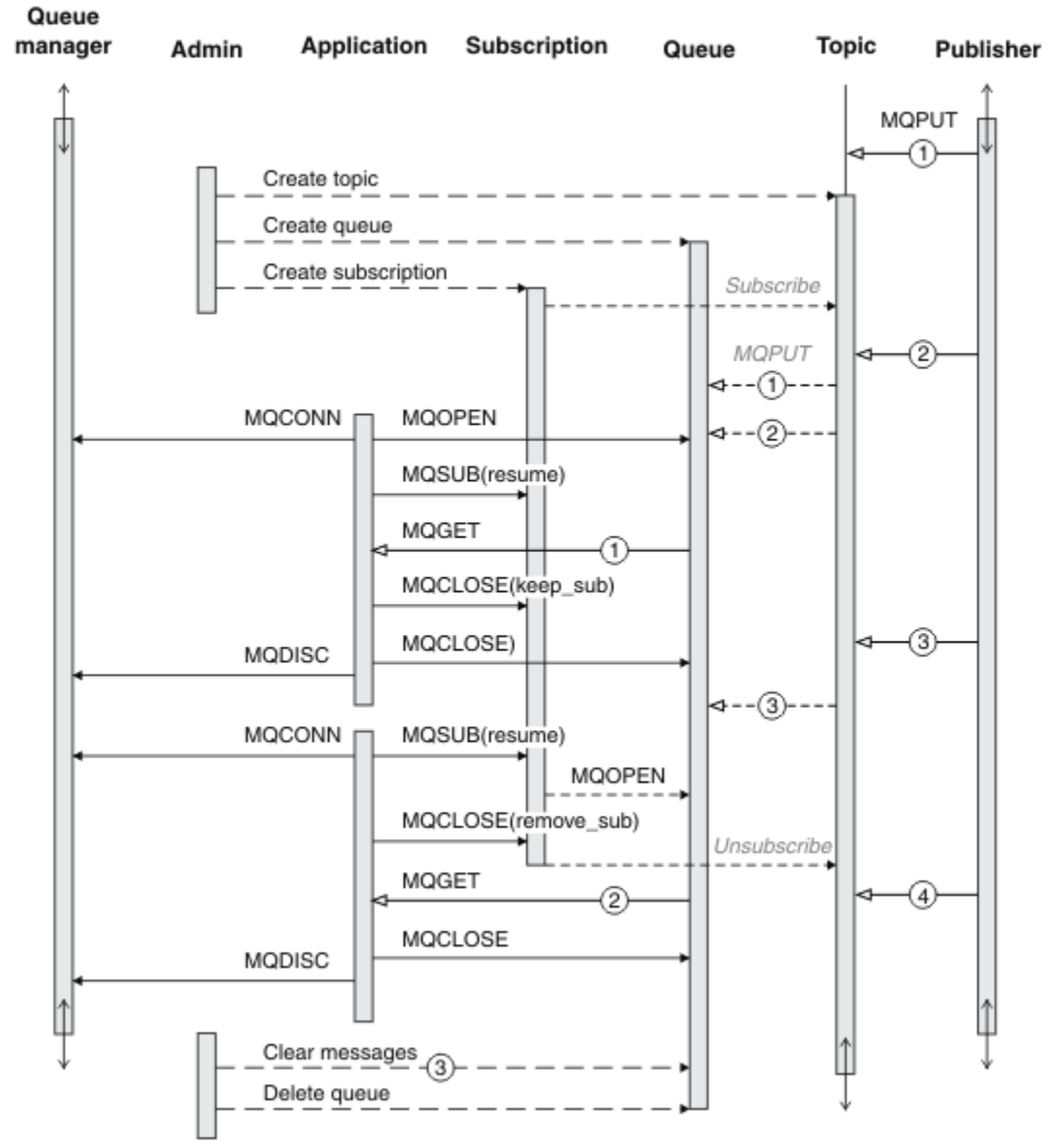
Ve třetím příkladu je přidán administrátor: nespravovaný trvalý odběratel. Jedná se o dobrý příklad, který ukazuje, jak může administrátor interaktivně spolupracovat s aplikací publikování/odběru.

Body, které je třeba poznamenat, jsou uvedeny.

1. Vydavatel vloží zprávu 1do tématu, které se později přidruží k objektu tématu používanému pro odběr. Objekt tématu definuje řetězec tématu, který odpovídá tématu publikovanému 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` popisovač fronty. Alternativně může jednoduše otevřít odběr a předat jej popisovači fronty `MQHO_NONE`. Konverzace není pravdivá, nemůže obnovit odběr předáním pouze popisovače fronty bez názvu odběru-fronta může mít více odběrů.
5. Aplikace otevře odběr pomocí volby `MQSO_RESUME` , i když se jedná o první otevření odběru. Pokračuje v administrativně vytvořeném odběru.
6. Odběratel obdrží zachované publikování 1. Publikace 2, ačkoli byla publikována před přijetím jakýchkoli publikací odběratelem, byla publikována po spuštění odběru a jedná se o druhé publikování ve frontě odběrů.

Poznámka: Není-li zachované publikování publikováno jako trvalá zpráva, bude 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 může administrátor udělat.
8. Výsledkem volby MQCO_REMOVE_SUB při uzavření odběru je odebrání odběru tak, jako by jej administrátor odstranil. Tím se zastaví odesílání dalších publikování do fronty, na rozdíl od *spravovaného* trvalého odběru však neovlivní publikování, která jsou již ve frontě, a to ani v případě, že je fronta zavřená.
9. Administrátor později odstraní zbývající zprávu 3a odstraní frontu.



Obrázek 85. Linie nespravovaných trvalých odběratelů

Běžným vzorem pro nespravovaný odběr je úklid front a odběrů, který má provádět administrátor. Obvykle se nepokoušíte emulovat chování spravovaného odběratele a uklizených front a odběrů programově v kódu aplikace. Pokud zjistíte, že potřebujete psát logiku správy, zpochybněte, zda můžete dosáhnout stejných výsledků pomocí spravovaného vzoru. Není snadné psát pevně synchronizovaný, zcela spolehlivý kód pro správu. Později, buď ručně, nebo pomocí automatizovaného programu správy, je snazší provést

uklizení, když si můžete být jisti, že zprávy, odběry a fronty lze jednoduše odstranit, bez ohledu na jejich stav.

Vlastnosti zprávy publikování/odběru

Několik vlastností zprávy se týká systému zpráv IBM MQ publikování/odběru.

Token PubAccounting

Jedná se o hodnotu, která bude v poli AccountingToken deskriptoru zpráv (MQMD) všech publikačních zpráv odpovídajících tomuto odběru. AccountingToken je součástí kontextu identity zprávy. Další informace o kontextu zprávy viz “kontext zprávy” na stránce 45. Další informace o poli AccountingToken v deskriptoru MQMD naleznete v tématu AccountingToken.

PubApplIdentityData

Jedná se o hodnotu, která bude v poli ApplIdentityData deskriptoru zprávy (MQMD) všech publikačních zpráv odpovídajících tomuto odběru. ApplIdentityData jsou součástí kontextu identity zprávy. Další informace o kontextu zprávy viz “kontext zprávy” na stránce 45. Další informace o poli dat ApplIdentityv deskriptoru MQMD viz ApplIdentityData.

Pokud není zadána volba MQSO_SET_IDENTITY_CONTEXT, jsou data ApplIdentity, která budou nastavena v každé zprávě publikované pro tento odběr, prázdná, jako výchozí informace o kontextu.

Je-li určena volba MQSO_SET_IDENTITY_CONTEXT, uživatel vygeneruje PubApplIdentityData a toto pole je vstupní pole obsahující data ApplIdentity, která mají být nastavena v každém publikování pro tento odběr.

PubPriority

Jedná se o hodnotu, která bude uvedena v poli Priorita deskriptoru zpráv (MQMD) všech zpráv publikování odpovídajících tomuto odběru. Další informace o poli Priorita v deskriptoru MQMD naleznete v tématu Priorita.

Hodnota musí být větší nebo rovna nule; nula je nejnižší priorita. Lze také použít následující speciální hodnoty:

- MQPRI_PRIORITY_AS_Q_DEF-Pokud je v poli Hobj ve volání MQSUB zadána fronta odběrů a nejedná se o spravovaný popisovač, bude priorita zprávy převzata z atributu DefProirity této fronty. Je-li takto identifikovaná fronta frontou klastru nebo existuje-li v cestě k rozlišení názvu fronty více než jedna definice, je priorita určena při vložení zprávy publikování do fronty, jak je popsáno v tématu Priorita v deskriptoru MQMD. Pokud volání MQSUB používá spravovaný popisovač, priorita zprávy je převzata z atributu DefPriority modelové fronty přidružené k odebíranému tématu.
- MQPRI_PRIORITY_AS_PUBLISHED-Prioritou zprávy je priorita původního publikování. Toto je počáteční hodnota tohoto pole.

SubCorrelId

Upozornění: Identifikátor korelace lze předávat pouze mezi správci front v klastru publikování/odběru, nikoli v hierarchii.

Všechna publikování odeslaná pro shodu s tímto odběrem budou obsahovat tento identifikátor korelace v deskriptoru zprávy. Pokud více odběrů používá stejnou frontu k získání svých publikací, použití MQGET podle ID korelace umožňuje získat pouze publikování pro konkrétní odběr. Tento identifikátor korelace může být generován správcem front nebo uživatelem.

Není-li zadána volba MQSO_SET_CORREL_ID, je identifikátor korelace generován správcem front a toto pole je výstupní pole obsahující identifikátor korelace, který bude nastaven v každé zprávě publikované pro tento odběr.

Pokud je zadána volba MQSO_SET_CORREL_ID, je identifikátor korelace generován uživatelem a toto pole je vstupní pole obsahující identifikátor korelace, který má být nastaven v každém publikování pro tento

odběr. V tomto případě, pokud pole obsahuje MQCI_NONE, bude identifikátorem korelace, který bude nastaven v každé zprávě publikované pro tento odběr, identifikátor korelace vytvořený původním vložením zprávy.

Je-li zadána volba MQSO_GROUP_SUB a zadaný identifikátor korelace je stejný jako existující seskupený odběr používající stejnou frontu a překrývající se řetězec tématu, bude s kopií publikování poskytnut pouze nejvýznamnější odběr ve skupině.

SubUserData

Jedná se o uživatelská data odběru. Data poskytnutá v rámci odběru v tomto poli budou zahrnuta jako vlastnost datové zprávy MQSubUserpro každé publikování odeslané v rámci tohoto odběru.

Vlastnosti publikování

Tabulka 124 na stránce 811 vypisuje vlastnosti publikování, které jsou poskytnuty se zprávou publikování.

K těmto vlastnostem můžete přistupovat přímo ze složky **MQRFH2** , nebo je můžete načíst pomocí MQINQMP. MQINQMP přijímá buď název vlastnosti, nebo název **MQRFH2** jako název vlastnosti, která se má dotazovat.

Tabulka 124. Vlastnosti publikování

Název vlastnosti	MQRFH2 název	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 publikování
MQPubTime	mqpse.Pts	MQTYPE_STRING	Čas publikování
MQPubSeqNum	mqpse.Seq	MQTYPE_INT32	Pořadové číslo publikace
MQPubStrIntData	mqpse.Sid	MQTYPE_STRING	Data typu String/Integer přidaná vydavatelem
MQPubFormat	mqpse.Pfmt	MQTYPE_INT32	Formát zprávy: MQRFH1 MQRFH2 PCF

Řazení zpráv

Pro konkrétní téma jsou zprávy publikovány správcem front ve stejném pořadí, v jakém jsou přijímány z publikačních aplikací (s výhradou změny pořadí na základě priority zpráv).

Řazení zpráv obvykle znamená, že každý odběratel přijímá zprávy od konkrétního správce front v konkrétním tématu od konkrétního vydavatele v pořadí, v jakém jsou publikovány tímto vydavatelem.

Avšak stejně jako u všech zpráv IBM MQ je možné, aby byly zprávy občas doručeny mimo pořadí. K tomu může dojít v následujících situacích:

- Pokud dojde k vypnutí propojení v síti a následné zprávy jsou přesměrovány na jiný spoj
- Dojde-li k dočasnému zaplnění fronty nebo k zablokování operace vložení, bude zpráva vložena do fronty nedoručených zpráv, a bude tedy zpožděna, zatímco následné zprávy budou procházet přímo.

- Pokud administrátor odstraní správce front v době, kdy vydavatelé a odběratelé stále pracují, způsobí, že zprávy ve frontě budou vloženy do fronty nedoručených zpráv a odběry budou přerušeny.

Pokud k těmto okolnostem nedojde, publikace jsou vždy dodávány v pořádku.

Poznámka: Nelze použít seskupené nebo segmentované zprávy s publikováním/odběrem.

Zachycování publikací

Můžete zachytit publikaci, upravit ji a poté ji znovu publikovat, než se dostane k jinému odběrateli.

Možná budete chtít zachytit publikování, než se dostane k odběrateli, abyste mohli provést jednu z následujících akcí:

- Připojit ke zprávě další informace
- Blokovat zprávu
- Transformovat zprávu

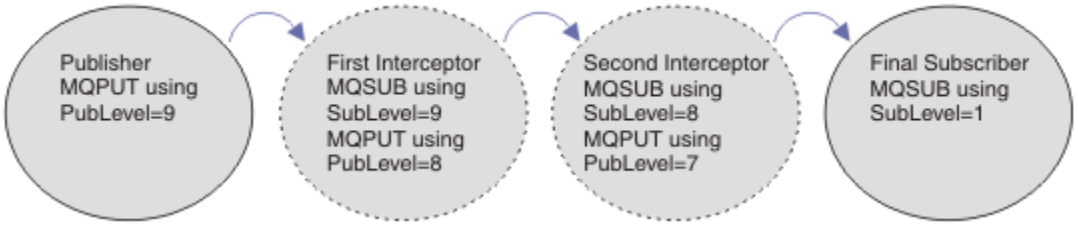
Pro každou zprávu můžete provést stejnou operaci nebo můžete operaci změnit v závislosti na odběru, zprávě nebo záhlaví zprávy.

Související odkazy

MQ_PUBLISH_EXIT-uživatelská procedura publikování

Úrovně odběru

Nastavte úroveň odběru tak, aby zachytila publikování před tím, než dosáhne konečných odběratelů. Zachycující odběratel se přihlásí k odběru na vyšší úrovni odběru a znovu publikuje na nižší úrovni publikování. Sestavte řetězec zachycených odběratelů, abyste provedli zpracování zpráv na publikování před jeho doručením konečným odběratelům.



Obrázek 86. Posloupnost zachycování odběratelů

Chcete-li zachytit publikování, použijte atribut **MQSD** `SubLevel` . Po zachycení zprávy ji lze transformovat a poté znovu publikovat na nižší úrovni publikování změnou atributu **MQPMO** `PubLevel` . Zpráva pak přejde na konečné odběratele, nebo je znovu zachycena přechodným odběratelem na nižší úrovni odběru.

Zachycující odběratel obvykle transformuje zprávu před jejím opětovným publikována. Posloupnost zachycování odběratelů vytváří tok zpráv. Alternativně nemusíte znovu publikovat zachycené publikování: Odběratelé na nižších úrovních odběru zprávu neobdrží.

Ujistěte se, že zachytávač přijímá publikování před všemi ostatními odběrateli. Nastavte úroveň odběru zachytávače vyšší než ostatní odběratelé. Standardně mají odběratelé `SubLevel` 1. Nejvyšší hodnota je 9. Publikování musí začínat řetězcem `PubLevel` alespoň na nejvyšší úrovni `SubLevel`. Publikujte na počátku s výchozí hodnotou `PubLevel` 9.

- Máte-li jednoho zachytávače v tématu, nastavte `SubLevel` na hodnotu 9.
- Pro více zachytávacích aplikací v tématu nastavte nižší úroveň `SubLevel` pro každého následného zachytávače.
- Můžete implementovat maximálně 8 zachycujících aplikací s úrovněmi odběru od 9 až po 2 včetně. Konečný příjemce zprávy má hodnotu `SubLevel` 1.

Zachytávač s nejvyšší úrovní odběru, která se rovná nebo je nižší než `PubLevel` publikování, obdrží nejprve publikování. Konfigurujte pouze jednoho zachytávače pro téma na konkrétní úrovni odběru. Máte-li více odběratelů na konkrétní úrovni odběru, bude odesláno více kopií publikování do konečné sady odebírajících aplikací.

Odběratel s úrovní SubLevel 0 se používá jako catchall. Obdrží publikování, pokud zprávu neobdrží žádný konečný odběratel. Odběratel s úrovní SubLevel 0 může být použit k monitorování publikování, která neobdrželi žádní jiní odběratelé.

Programování zachytávače

Použijte volby odběru popsané v části Tabulka 125 na stránce 813.

Tabulka 125. Volby odběru pro zachytávání odběratelů

Volba odběru	**Notes**
MQSO_SET_CORREL_ID a SubCorrelSubCorrel nastaveno na MQCI_NONE	Ponechte CorrelId zachyceného publikování stejné jako původní publikování. **Poznámka:** Nelze předat identifikátor korelace publikování v hierarchii. Pole používá správce front.
PubPriority nastavit na MQPRI_PRIORITY_AS_PUBLISHED	Zachovejte prioritu zachycené publikace stejnou jako původní publikace.

Volby v souboru Tabulka 125 na stránce 813 musí používat všichni zachycující odběratelé. Výsledkem je, že identifikátor korelace a priorita zprávy nejsou upraveny z nastavení původního vydavatele.

Po zpracování publikování zachytávajícím odběratelem znovu publikuje zprávu ve stejném tématu na úrovni PubLevel nižší než úroveň SubLevel svého vlastního odběru. Pokud zachycující odběratel nastaví SubLevel 9, znovu publikuje zprávu s PubLevel 8.

Chcete-li zprávu znovu publikovat správně, je třeba zadat několik informací z původní publikace. Znovu použijte stejný parametr **MQMD** jako v původní zprávě a nastavte parametr MQPMO_PASS_ALL_CONTEXT , abyste zajistili, že všechny informace v souboru **MQMD** budou předány dalšímu odběrateli. Zkopírujte hodnoty z vlastností zprávy zobrazených v souboru Tabulka 126 na stránce 813 do příslušných polí znovu publikovanou zprávu. Zachycující odběratel může tyto hodnoty změnit. Pomocí operátoru OR přidejte další hodnoty do **MQPMO**. Pole Volby , chcete-li kombinovat volby vložení zprávy.

Frontu publikování je třeba otevřít explicitně, nikoli použít spravovanou frontu publikování. Pro spravovanou frontu nelze nastavit hodnotu MQSO_SET_CORREL_ID . Také nemůžete nastavit MQOO_SAVE_ALL_CONTEXT ve spravované frontě. Viz fragmenty kódu uvedené v seznamu "Příklady" na stránce 814.

Tabulka 126. Hodnoty MQPUT pro opětné publikování zpráv

Znovu publikovat zprávu pomocí příkazu MQPUT	**Informace v publikační zprávě**
MQOD. ObjectString	vlastnost zprávy MQTopicString
MQPMO. Options	vlastnost zprávy MQPubOptions

Konečný odběratel má možnost nastavit své volby odběru odlišně. Může například nastavit prioritu publikování explicitně a nikoli na hodnotu MQPRI_PRIORITY_AS_PUBLISHED. Nastavení konečného odběratele ovlivní pouze publikování z konečného zachytávaného odběratele v řetězci.

Zachovaná publikování

Zachované publikování musí být po zachycení zachováno zkopírováním původních voleb vložení zprávy do znovu publikovanou zprávy.

Volba MQPMO_RETAIN je nastavena vydavatelem. Každý zachycující odběratel musí přenést soubor MQPubOptions do voleb vložení zprávy pro znovu publikovanou zprávu, jak je uvedeno v části Tabulka 126 na stránce 813. Zkopírování voleb vložení zprávy zachová volby nastavené původním vydavatelem, včetně toho, zda má být zachováno publikování.

Když publikace dokončí svůj průchod řetězcem zachycování odběratelů a je doručena konečným odběratelům, je nakonec zachována. Noví odběratelé na `SubLevel` 1požadující zachované publikování obdrží toto publikování bez dalšího odposlechu. Odběratelům na `SubLevel` vyšší než 1 se zachované publikování neodešle. Výsledkem je, že zachované publikování není upraveno řetězcem zachycování odběratelů v druhém kole.

Příklady

Příklady jsou fragmenty kódu, které lze kombinovat za účelem sestavení zachytávacího odběratele. Kód je napsán tak, aby byl stručný, spíše než o kvalitě výroby.

Direktivy preprocesoru v souboru Obrázek 87 na stránce 814 definují dvě vlastnosti, které mají být extrahovány ze zpráv publikování vyžadovaných voláním rozhraní `MQINQMP` MQI.

```
#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 87. Direktivy preprocesoru

Obrázek 88 na stránce 815 uvádí deklarace použité ve fragmentech kódu. S výjimkou zvýrazněných výrazů jsou deklarace pro aplikaci IBM MQ standardní.

Zvýrazněné volby Put a Get jsou inicializovány tak, aby předaly veškerý kontext. Zvýrazněné `MQTOPICSTRING` a `MQPUBOPTIONS` jsou inicializátory MQCHARV pro názvy vlastností, které jsou definovány v direktivách preprocesoru. Názvy se předávají do adresáře `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 88. Prohlášení

Inicializace, které nelze snadno provést v deklaracích, jsou zobrazeny v souboru Obrázek 89 na stránce 816. Zvýrazněné hodnoty vyžadují vysvětlení.

SYSTEM.NDURABLE.MODEL.QUEUE
: V tomto příkladu se namísto použití MQSUB k otevření spravovaného dočasného odběru používá modelová fronta SYSTEM.NDURABLE.MODEL.QUEUEk vytvoření dočasné dynamické fronty. Jeho popisovač je předán do adresáře MQSUB. Otevřením fronty přímo můžete uložit veškerý kontext zprávy a nastavit volbu odběru MQSO_SET_CORREL_ID.

MQGMO_CURRENT_VERSION
: Je důležité použít aktuální verzi většiny struktur IBM MQ . Pole jako gmo.MsgHandle jsou k dispozici pouze v nejnovější verzi řídicích struktur.

MQGMO_PROPERTIES_IN_HANDLE
: Řetězec tématu a volby vložení zprávy nastavené v původním publikování mají být načteny zachytávajícím odběratelem pomocí vlastností zprávy. Alternativou by bylo číst strukturu **MQRFH2** ve zprávě přímo.

MQSO_SET_CORREL_ID
: Použijte MQSO_SET_CORREL_ID v kombinaci s,

```
memcpy(sd.SubCorrelId, MQCI_NONE, sizeof(sd.SubCorrelId));
```

Výsledkem těchto voleb je předání identifikátoru korelace. Identifikátor korelace nastavený původním vydavatelem je umístěn do pole identifikátoru korelace publikování, které je přijato zachycujícím odběratelem. Každý zachycující odběratel předává stejný identifikátor korelace. Konečný odběratel pak má možnost přijmout stejný identifikátor korelace.

Poznámka: Pokud je publikování předáno prostřednictvím hierarchie publikování/odběru, identifikátor korelace se nikdy nezachová.

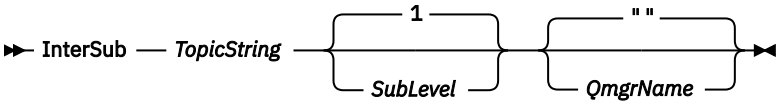
MQPRI_PRIORITY_AS_PUBLISHED
Publikování je umístěno do fronty publikování se stejnou prioritou zprávy, s jakou bylo publikováno.

```
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 89. Inicializace

Obrázek 90 na stránce 817 zobrazuje fragment kódu pro čtení parametrů příkazového řádku, dokončení inicializace a vytvoření zachytávacího odběru.

Spusťte program pomocí příkazu,



Aby bylo ošetření chyb co nejnenápadnější, je kód příčiny z každého volání MQI uložen v jiném prvku pole. Po každém volání je otestován kód dokončení a pokud je hodnota MQCC_FAIL, řízení ukončí blok kódu do `{ } while(0)`.

Dva pozoruhodné řádky kódu jsou:

pmo.PubLevel = sd.SubLevel - 1;
Nastaví úroveň publikování pro znovu publikovanou zprávu na nižší úroveň, než je úroveň odběru zachytávaného odběratele.

gmo.MsgHandle = Hmsg;
Poskytuje popisovač zprávy pro MQGET k vrácení vlastností 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 90. Příprava na zachycení publikací

Fragment hlavního kódu Obrázek 91 na stránce 818získává zprávy z fronty publikování. Dotazuje se na vlastnosti zprávy a znovu publikuje zprávy pomocí řetězce tématu a původního souboru **MQPMO**. `option` vlastnosti publikování.

V tomto příkladu se na publikaci neprovádí žádná transformace. Řetězec tématu znovu publikovaný se vždy shoduje s řetězcem tématu, na kterém se zachycoval odběratel, který byl přihlášen k odběru. Pokud zachycující odběratel odpovídá za zachycení více odběrů odeslaných do stejné fronty publikování, může být nutné zadat dotaz na řetězec tématu, aby bylo možné rozlišit publikování, která se shodují s různými odběry.

Volání do `MQINQMP` jsou zvýrazněna. Vlastnosti řetězce tématu a volby vložení publikování jsou zapsány přímo do řídicích struktur výstupu. Jedinou příčinou změny pole délky MQCHARV `putOD.ObjectString` z explicitní délky na řetězec ukončený hodnotou null je použití `printf` pro výstup řetězce.

```
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 91. Zachytit publikaci a znovu publikovat

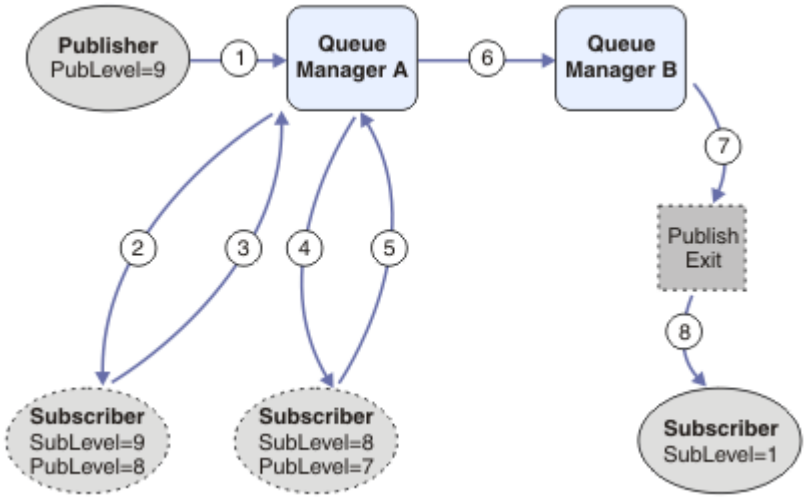
Finální fragment kódu je zobrazen v souboru Obrázek 92 na stránce 818.

```
    } 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 92. Dokončení

Zachytávání publikací a distribuovaného publikování/odběru
Při implementaci zachytávacích odběratelů nebo uživatelských procedur publikování v distribuované topologii publikování/odběru postupujte podle jednoduchého vzoru. Implementujte zachytávače ve stejných správcích front jako vydavatelé a uživatelské procedury publikování ve stejných správcích front jako koncoví odběratelé.

V části Obrázek 93 na stránce 819 jsou uvedeni dva správci front připojení v klastru publikování a odběrů. Vydavatel vytvoří publikování do tématu klastru na úrovni publikování 9. Číslované šipky ukazují posloupnost kroků, které publikace podniká při průtoku k odběratelům tématu klastru. Publikování je zachycováno odběratelem s volbou `Dílčí úroveň` 9 a znovu publikován s volbou `Publevel` 8. Je znovu zachycen odběratelem na `dílčí úrovni` 8. Odběratel znovu publikuje na úrovni `Publevel` 7. Proxy odběratel poskytovaný správcem front předává publikování správci front B, kde byla kromě konečného odběratele implementována i uživatelská procedura publikování. Publikování je zpracováno uživatelskou procedurou pro publikování, než je nakonec přijato konečným odběratelem na `dílčí úrovni` 1. Zachycující odběratelé a uživatelská procedura publikování jsou zobrazeny s přerušenými osami.



Obrázek 93. Ukončení zachycení a publikování v klastru

Cílem jednoduchého vzoru je, aby každý odběratel, který obdrží publikování, obdržel stejné publikování. Publikování prochází stejnou posloupností transformací bez ohledu na to, kde je odběratel připojen. Pravděpodobně se chcete vyhnout tomu, aby se posloupnost transformací měnila v závislosti na tom, kde jsou připojeni vydavatelé nebo koncoví odběratelé. Rozumnou výjimkou by bylo přizpůsobení publikace nakonec doručené každému jednotlivému účastníkovi. Pomocí uživatelské procedury publikování upravte publikování na základě fronty, do které je publikování nakonec doručeno.

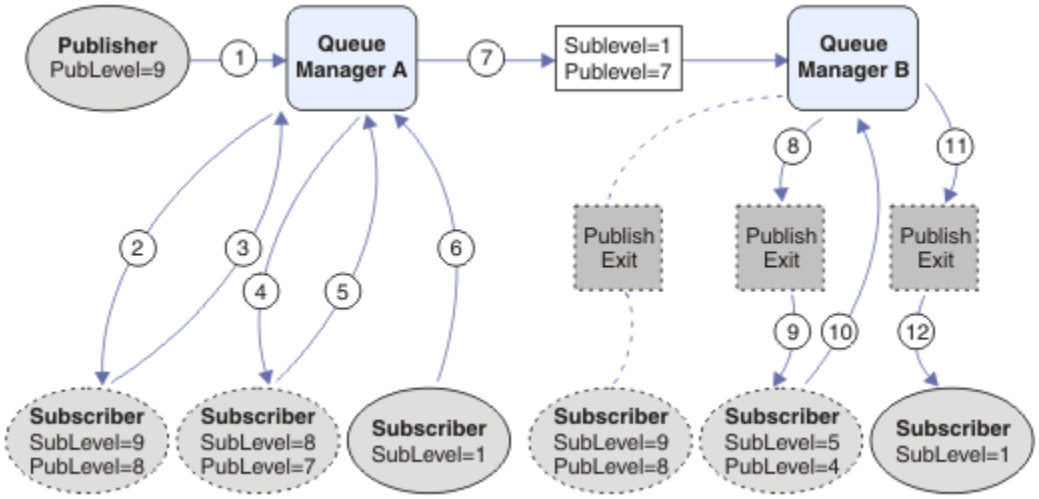
Musíte pečlivě zvážit, kam implementovat zachycující odběratele a uživatelské procedury publikování v distribuované topologii publikování/odběru. Přímý vzor implementuje zachycující odběratele do stejného správce front jako vydavatelé a uživatelské procedury Publikovat do stejných správců front jako koncoví odběratelé.

Anti-vzor

Obrázek 94 na stránce 820 ukazuje, jak věci mohou jít awry, pokud nechcete postupovat podle jednoduchého vzoru. Aby se implementace zkomplikovala, je do správce front A přidán konečný odběratel a do správce front B jsou přidáni dva další zachycující odběratelé.

Publikování je předáno správci front B na adrese `PubLevel` 7, kde je zachycováno odběratelem na `SubLevel` 5 před tím, než je spotřebováno konečným odběratelem na `SubLevel` 1. Uživatelská procedura publikování zachytí publikování před jeho předáním zachycující odběrateli i konečnému odběrateli ve správci front B. Publikování dosáhne konečného odběratele ve správci front A, aniž by bylo zpracováno uživatelskou procedurou pro publikování.

V topologii publikování/odběru se odběratelé proxy přihlašují k odběru na úrovni `SubLevel` 1a předávají `PubLevel` nastavené posledním zachycujícím odběratelem. V produktu Obrázek 94 na stránce 820je výsledkem to, že publikování není zachycováno odběratelem pomocí `SubLevel` 9 ve správci front B.



Obrázek 94. Komplexní implementace zachytávacích odběratelů

Volby publikování

K dispozici je několik voleb, které řídí způsob publikování zpráv.

Neposkytnutí informací o odpovědi odběratelům

Pokud nechcete, aby mohli odběratelé odpovídat na přijímaná publikování, je možné zadržet informace v polích ReplyToQ a ReplyTosprávce front deskriptoru MQMD pomocí volby MQPMO_SUPPRESS_REPLYTO put-message. Je-li použita tato volba, správce front odebere tyto informace z deskriptoru MQMD při přijetí publikování před jejich předáním všem odběratelům.

Tuto volbu nelze použít v kombinaci s volbou sestavy, která vyžaduje Q ReplyTo, pokud se jedná o pokus o volání s chybou MQRC_MISSING_REPLY_TO_Q.

Úroveň publikování

Použití úrovní publikování je způsob, jak řídit, kteří odběratelé obdrží publikování. Úroveň publikování označuje úroveň odběru, na kterou je publikování zaměřeno. Publikování obdrží pouze odběry s nejvyšší úrovní odběru nižší nebo rovnou úrovni publikování publikování. Tato hodnota musí být v rozsahu od nuly do devíti; nula je nejnižší úroveň publikování. Počáteční hodnota tohoto pole je 9. Jednou z možností použití úrovní publikování a odběrů je zachytávat publikování.

Kontrola, zda není publikování doručeno žádnému odběrateli

Chcete-li zkontrolovat, zda publikování nebylo doručeno žádnému odběrateli, použijte volbu MQPMO_WARN_IF_NO_SUBS_MATCHED put-message s voláním MQPUT. Pokud operace vložení vrátí kód dokončení MQCC_WARNING a kód příčiny MQRC_NO_SUBS_MATCHED, publikování nebylo doručeno žádným odběrům. Je-li v operaci vložení určena volba MQPMO_RETAIN, zpráva se zachová a doručí do libovolného následně definovaného odpovídajícího odběru. V distribuovaném systému publikování/odběru je kód příčiny MQRC_NO_SUBS_MATCHED vrácen pouze v případě, že pro téma ve správci front nejsou registrovány žádné proxy odběry.

Volby odběru

K dispozici je několik voleb, které řídí způsob zpracování odběrů zpráv.

Trvalost zpráv

Správci front udržují perzistenci publikací, které předávají odběratelům podle nastavení vydavatele. Vydavatel nastaví perzistenci na jednu z následujících voleb:

0
Dočasný

1
Trvalý

2
Perzistence jako definice fronty/tématu

V případě publikování/odběru vydavatel interpretuje objekt tématu a **topicString** na vyřešený objekt tématu. Pokud vydavatel určí perzistenci jako definici fronty/tématu, bude pro publikování nastavena výchozí perzistence z vyřešeného objektu tématu.

Zachovaná publikování

Chcete-li řídit, kdy jsou zachovaná publikování přijímána, mohou odběratelé použít dvě volby odběru:

Publikovat pouze na vyžádání, MQSO_PUBLICATIONS_ON_REQUEST

Chcete-li, aby měl odběratel kontrolu nad tím, kdy obdrží publikování, můžete použít volbu odběru MQSO_PUBLICATIONS_ON_REQUEST. Odběratel pak může řídit, kdy obdrží publikování, pomocí volání MQSUBRQ (s uvedením manipulátoru Hsub, který byl vrácen z původního volání MQSUB), a požadovat, aby mu bylo odesláno zachované publikování tématu. Odběratelé používající volbu odběru MQSO_PUBLICATIONS_ON_REQUEST nepřijímají žádná nezachovaná publikování.

Zadáte-li MQSO_PUBLICATIONS_ON_REQUEST, musíte k načtení publikování použít MQSUBRQ. Pokud parametr MQSO_PUBLICATIONS_ON_REQUEST nepoužijete, obdržíte zprávy při jejich publikování.

Pokud odběratel používá volání MQSUBRQ a v tématu odběru používá zástupné znaky, může odběr odpovídat více tématům nebo uzlům ve stromu témat, přičemž všechny z nich se zachovanými zprávami (existují-li) budou odesílány odběrateli.

Tato volba může být užitečná zejména v případě použití s trvalými odběry, protože správce front bude nadále odesílat publikování odběrateli, pokud se k odběru trvale přihlásí i v případě, že aplikace odběratele není spuštěna. To může vést k nahromadění zpráv ve frontě odběratele. Tomuto sestavení se lze vyhnout, pokud se odběratel registruje pomocí volby MQSO_PUBLICATIONS_ON_REQUEST. Případně můžete použít dočasné odběry, pokud je to vhodné pro vaši aplikaci, abyste se vyhnuli hromadění nežádoucích zpráv.

Pokud je odběr trvalý a vydavatel používá zachovaná publikování, může aplikace odběratele použít volání MQSUBRQ k aktualizaci informací o stavu po restartování. Odběratel pak musí pravidelně aktualizovat svůj stav pomocí volání MQSUBRQ.

V důsledku volání MQSUB s použitím této volby nebudou odeslána žádná publikování. Trvalý odběr, který byl obnoven po odpojení, bude používat volbu MQSO_PUBLICATIONS_ON_REQUEST, pokud byl původní odběr konfigurován pro použití této volby.

Pouze nová publikování, MQSO_NEW_PUBLICATIONS_ONLY

Pokud v tématu existuje zachované publikování, obdrží každý odběratel, který provede odběr po provedení publikování, kopii tohoto publikování. Pokud odběratel nechce přijímat žádná publikování, která byla vytvořena dříve než daný odběr, může použít volbu odběru MQSO_NEW_PUBLICATIONS_ONLY.

Seskupování odběrů

Seskupování odběrů zvažte v případě, že jste nastavili frontu pro příjem publikování a máte k dispozici řadu překrývajících se odběrů, které zásobují publikování do stejné fronty. Tato situace je podobná příkladu v tématu Překrývající se odběry.

Můžete se vyhnout příjmu duplicitních publikování nastavením volby MQSO_GROUP_SUB při přihlášení k odběru tématu. Výsledkem je, že pokud se více než jeden odběr ve skupině shoduje s tématem publikování, odpovídá za umístění publikování do fronty pouze jeden odběr. Ostatní odběry, které odpovídají tématu publikování, jsou ignorovány.

Odběr odpovědný za umístění publikování do fronty je vybrán na základě toho, že má nejdelší odpovídající řetězec tématu, než narazí na jakékoli zástupné znaky. To lze považovat za nejbližší odpovídající předplatné. Jeho vlastnosti jsou šířeny do publikování, včetně toho, zda má vlastnost MQSO_NOT_OWN_PUBS . Pokud ano, do fronty se nedoručí žádné publikování, i když jiné odpovídající odběry nemusí mít vlastnost MQSO_NOT_OWN_PUBS .

Nemůžete umístit všechny své odběry do jedné skupiny, abyste eliminovali duplicitní publikování. Seskupené odběry musí splňovat tyto podmínky:

1. Žádné odběry nejsou spravovány.
2. Skupina odběrů doručuje publikování do stejné fronty.
3. Každý odběr musí být na stejné úrovni odběru.
4. Zpráva publikování pro každý odběr ve skupině má stejný identifikátor korelace.

   Chcete-li zajistit, aby každý odběr vytvořil zprávu publikování se stejným identifikátorem korelace, nastavte volbu MQSO_SET_CORREL_ID tak, aby byl v publikování vytvořen vlastní identifikátor korelace, a nastavte v každém odběru stejnou hodnotu v poli **SubCorrelId** . Nenastavujte parametr **SubCorrelId** na hodnotu MQCI_NONE.

Další informace viz ../refdev/q100080_.dita#q100080_/mqso_group_sub .

Zjišťování a nastavení atributů objektu

Atributy jsou vlastnosti, které definují charakteristiky objektu IBM MQ .

Ovlivňují způsob, 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 změnit pouze pomocí příkazů IBM MQ ; příkladem takového atributu je výchozí priorita pro zprávy vkládané do fronty. Ostatní atributy jsou ovlivněny operací správce front a mohou se v průběhu času měnit; příkladem je aktuální hloubka fronty.

Pomocí volání MQINQ můžete zjistit aktuální hodnoty většiny atributů. Rozhraní MQI také poskytuje volání MQSET, pomocí kterého můžete změnit některé atributy fronty. Volání MQI nelze použít ke změně atributů jakéhokoli jiného typu objektu. Místo toho musíte použít jeden z následujících prostředků:

- ALW Zařízení MQSC, které je popsáno v tématu Příkazy MQSC.
- IBM i CL příkazy CHGMQMx, které jsou popsány v příručce CL commands for IBM i, nebo v zařízení MQSC.
- z/OS Příkazy ALTER nebo DEFINE s volbou REPLACE, které jsou popsány v části Příkazy MQSC.

Poznámka: Názvy atributů objektů jsou uvedeny v této dokumentaci ve formě, kterou používáte s voláními MQINQ a MQSET. Když použijete příkazy IBM MQ k definování, změně nebo zobrazení atributů, musíte tyto atributy identifikovat pomocí klíčových slov zobrazených v popisech příkazů v odkazech na téma.

Volání MQINQ i MQSET používají k identifikaci pole selektorů. ty atributy, které chcete dotazovat nebo nastavit. Pro každý atribut, se kterým můžete pracovat, existuje selektor. Název selektoru má předponu určenou povahou atributu:

Tabulka 127. Předpony pro názvy selektorů

Předpona	**Popis**
MQCA_	Tyto selektory odkazují na atributy, které obsahují znaková data (například název fronty).

Tabulka 127. Předpony pro názvy selektorů (pokračování)	
Předpona	**Popis**
MQIA_	Tyto selektory odkazují na atributy, které obsahují buď číselné hodnoty (například *CurrentQueueDepth*, počet zpráv ve frontě), nebo konstantní hodnotu (například *SyncPoint*, zda správce front podporuje synchronizační body).

Před použitím volání MQINQ nebo MQSET musí být aplikace připojena ke správci front a musíte použít volání MQOPEN k otevření objektu pro nastavení nebo dotazování na atributy. Tyto operace jsou popsány v části "Připojení ke správci front a odpojení od něj" na stránce 713 a "Otevírání a zavírání objektů" na stránce 720.

Pomocí následujících odkazů se dozvíte více o dotazování a nastavení atributů objektů:

- "Zjišťování atributů objektu" na stránce 823
- "Některé případy, kdy volání MQINQ selže" na stránce 824
- "Nastavení atributů fronty" na stránce 825

Související pojmy

"Přehled rozhraní fronty zpráv" na stránce 700
Informace o komponentách rozhraní MQI (Message Queue Interface).

"Připojení ke správci front a odpojení od něj" na stránce 713
Chcete-li používat programovací služby IBM MQ , musí mít program připojení ke správci front. Pomocí těchto informací se dozvíte, jak se připojit ke správci front a jak se od něj odpojit.

"Otevírání a zavírání objektů" na stránce 720
Tyto informace poskytují náhled na otevírání a zavírání objektů IBM MQ .

"Vkládání zpráv do fronty" na stránce 730
Pomocí těchto informací se dozvíte, jak vkládat zprávy do fronty.

"Získávání zpráv z fronty" na stránce 744
Pomocí těchto informací získáte informace o získávání zpráv z fronty.

"Potvrzení a zálohování jednotek práce" na stránce 825
Tyto informace popisují, jak potvrdit a vrátit zpět všechny obnovitelné operace get a put, které se vyskytly v transakci.

"Spuštění aplikací IBM MQ pomocí spouštěčů" na stránce 836
Získejte informace o spouštěčích a o tom, jak spouštět aplikace IBM MQ pomocí spouštěčů.

"Práce s rozhraním MQI a klastry" na stránce 854
Existují speciální volby pro volání a návratové kódy, které se týkají klastrování.

"Používání a psaní aplikací na systému IBM MQ for z/OS" na stránce 858
Aplikace IBM MQ for z/OS lze vytvořit z programů, které jsou spuštěny v mnoha různých prostředích. To znamená, že mohou využívat zařízení, která jsou k dispozici ve více než jednom prostředí.

"IMS a IMS přemostění aplikací na IBM MQ for z/OS" na stránce 66
Tyto informace vám pomohou psát aplikace IMS pomocí IBM MQ.

Zjišťování atributů objektu

Pomocí volání MQINQ můžete zjistit atributy libovolného typu IBM MQ.

Jako vstup pro toto volání musíte zadat:

- Manipulátor připojení.
- Popisovač objektu.
- Počet selektorů.

- Pole selektorů atributů, každý selektor má tvar MQCA_ * nebo MQIA_ *. Každý selektor představuje atribut s hodnotou, na kterou se chcete dotazovat, a každý selektor musí být platný pro typ objektu, který představuje popisovač objektu. Selektory můžete zadat v libovolném pořadí.
- Počet celočíselných atributů, na které se dotazujete. Pokud se nedotazujete na celočíselné atributy, zadejte hodnotu nula.
- Délka vyrovnávací paměti atributů znaků v souboru *CharAttrLength*. Musí se jednat alespoň o součet délek požadovaných pro uchování každého řetězce znakového atributu. Uveďte nulu, pokud se nedotazujete na znakové atributy.

Výstup z MQINQ je:

- Sada celočíselných hodnot atributů zkopírovaných do pole. Počet hodnot je určen hodnotou *IntAttrCount*. Má-li parametr *IntAttrCount* nebo *SelectorCount* hodnotu nula, tento parametr se nepoužije.
- Vyrovnávací paměť, ve které jsou vráceny atributy znaků. Délka vyrovnávací paměti je dána parametrem **CharAttrLength** . Má-li parametr *CharAttrLength* nebo *SelectorCount* hodnotu nula, tento parametr se nepoužije.
- Kód dokončení. Pokud kód dokončení vydá varování, znamená to, že volání bylo dokončeno pouze částečně. V tomto případě zkontrolujte kód příčiny.
- Kód příčiny. Existují tři situace částečného dokončení:
  - Selektor se nevztahuje na typ fronty.
  - Pro celočíselné atributy není dostatek prostoru.
  - Pro znakové atributy není povolen dostatek prostoru.

  Pokud nastane více než jedna z těchto situací, vrátí se první situace, která se použije.

Pokud otevřete frontu pro výstup nebo dotaz a ta se vyřeší na jinou než lokální frontu klastru, můžete se dotázat pouze na název fronty, typ fronty a společné atributy. Hodnoty společných atributů jsou ty z vybrané fronty, pokud byla použita hodnota MQOO_BIND_ON_OPEN. Jedná se o hodnoty libovolné jedné z možných front klastru, pokud byla použita hodnota 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 855 a MQOPEN .

Poznámka: Hodnoty vrácené voláním jsou snímkem vybraných atributů. Atributy se mohou změnit dříve, než váš program provede činnost na vrácených hodnotách.

Existuje popis volání MQINQ v MQINQ.

Některé případy, kdy volání MQINQ selže

Pokud otevřete alias pro dotazování na jeho atributy, vrátí se atributy alias fronty (objekt IBM MQ používaný pro přístup k jiné frontě), nikoli atributy základní fronty.

Avšak definice základní fronty, na kterou se alias interpretuje, 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, pokud se změní atributy objektu alias fronty.

Podobně, když otevřete vzdálenou frontu, abyste se dotázali na její atributy, vrátí se vám pouze atributy lokální definice vzdálené fronty.

Zadáte-li jeden nebo více selektorů, které nejsou platné pro typ atributů fronty, na které se dotazujete, volání MQINQ se dokončí s varováním a nastaví výstup takto:

- V případě celočíselných atributů jsou odpovídající prvky *IntAttrs* nastaveny na hodnotu MQIAV_NOT_POUŽITELNÉ.
- Pro znakové atributy jsou odpovídající části řetězce *CharAttrs* nastaveny na hvězdičky.

Zadáte-li jeden nebo více selektorů, které nejsou platné pro typ atributů objektu, na kterém se dotazujete, volání MQINQ se nezdaří a vrátí kód příčiny MQRC_SELECTOR_ERROR.

Nemůžete volat MQINQ pro vyhledání modelové fronty; použijte buď prostředek MQSC, nebo příkazy dostupné na vaší platformě.

Nastavení atributů fronty

Pomocí těchto informací se dozvíte, jak nastavit atributy fronty pomocí volání MQSET.

Pomocí volání MQSET lze nastavit pouze následující atributy fronty:

- *InhibitGet* (ale ne pro vzdálené fronty)
- *DistList* (není zapnuto z/OS)
- *InhibitPut*
- *TriggerControl*
- *TriggerType*
- *TriggerDepth*
- *TriggerMsgPriority*
- *TriggerData*

Volání MQSET má stejné parametry jako volání MQINQ. Pro MQSET jsou však všechny parametry kromě kódu dokončení a kódu příčiny vstupními parametry. Neexistují žádné situace částečného dokončení.

Poznámka: Rozhraní MQI nelze použít k nastavení atributů jiných objektů IBM MQ než lokálně definovaných front.

Další podrobnosti o volání MQSET viz MQSET.

Potvrzení a zálohování jednotek práce

Tyto informace popisují, jak potvrdit a vrátit zpět všechny obnovitelné operace get a put, které se vyskytly v transakci.

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 přeskočit na “Aspekty synchronizačního bodu v aplikacích IBM MQ” na stránce 827.

Potvrdit a vrátit zpět

Když program vloží zprávu do fronty v rámci pracovní jednotky, tato zpráva se ostatním programům zviditelní pouze tehdy, když program potvrdí pracovní jednotku. Chcete-li potvrdit jednotku práce, musí být všechny aktualizace úspěšné, aby byla zachována integrita dat. Pokud program zjistí chybu a rozhodne se, že operace vložení není trvalá, může jednotku práce vrátit zpět. Když program provede odvolání, produkt IBM MQ obnoví frontu odebráním zpráv, které byly vloženy do fronty touto pracovní jednotkou. Způsob, jakým program provádí operace commit a back-out, závisí na prostředí, ve kterém je program spuštěn.

Podobně, když program získá zprávu z fronty v rámci pracovní jednotky, tato zpráva zůstane ve frontě, dokud program nepotvrdí pracovní jednotku, ale tato zpráva není k dispozici pro načtení jinými programy. Zpráva je trvale odstraněna z fronty, když program potvrdí pracovní jednotku. Pokud program odvolá 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, jehož prostřednictvím jsou pracovní jednotky potvrzeny nebo zálohovány s integritou dat.

Rozhodnutí o potvrzení nebo vrácení změn je učiněno v nejjednodušším případě na konci transakce. Může však být užitečnější, aby aplikace synchronizovala změny dat v jiných logických bodech v rámci transakce. Tyto logické body se nazývají *synchronizační body* (nebo *synchronizační body*). a období zpracování sady aktualizací mezi dvěma synchronizovanými body se nazývá *jednotka práce*. Součástí jedné pracovní jednotky může být několik volání MQGET a volání MQPUT.

Maximální počet zpráv v rámci pracovní jednotky lze řídit atributem MAXUMSGS příkazu ALTER QMGR .

jednofázové potvrzení

Proces *jednofázového potvrzování* je proces, v němž může program potvrdit aktualizace fronty bez koordinace změn s ostatními správci prostředků.

Dvoufázové potvrzení

Proces *dvoufázového potvrzování* je proces, při kterém lze aktualizace, které program provedl ve frontách systému IBM MQ , koordinovat s aktualizacemi jiných prostředků (například databází řízených produktem Db2). V rámci takového procesu jsou aktualizace všech prostředků potvrzeny nebo zálohovány společně.

Produkt IBM MQ poskytuje atribut **BackoutCount** , který vám pomůže zpracovat pracovní jednotky. Tato hodnota se zvýší pokaždé, když se zpráva v rámci pracovní jednotky odvolá. Pokud zpráva opakovaně způsobí nestandardní ukončení jednotky práce, hodnota parametru *BackoutCount* nakonec překročí hodnotu parametru *BackoutThreshold*. Tato hodnota je nastavena při definování fronty. V této situaci může aplikace odebrat zprávu z pracovní jednotky a vložit ji do jiné fronty, jak je definováno v *BackoutRequeueQName*. Když je zpráva přesunuta, jednotka práce může potvrdit.

Pomocí následujících odkazů se dozvíte více o potvrzování a zálohování jednotek práce:

- “Aspekty synchronizačního bodu v aplikacích IBM MQ” na stránce 827
- z/OS “Synchronizační body v aplikacích IBM MQ for z/OS” na stránce 828
- IBM i “Synchronizační body v CICS pro aplikace IBM i” na stránce 830
- “Synchronizační body v adresáři IBM MQ for Multiplatforms” na stránce 830
- IBM i “Rozhraní pro externího správce synchronizačního bodu IBM i” na stránce 835

Související pojmy

“Přehled rozhraní fronty zpráv” na stránce 700
Informace o komponentách rozhraní MQI (Message Queue Interface).

“Připojení ke správci front a odpojení od něj” na stránce 713
Chcete-li používat programovací služby IBM MQ , musí mít program připojení ke správci front. Pomocí těchto informací se dozvíte, jak se připojit ke správci front a jak se od něj odpojit.

“Otevírání a zavírání objektů” na stránce 720
Tyto informace poskytují náhled na otevírání a zavírání objektů IBM MQ .

“Vkládání zpráv do fronty” na stránce 730
Pomocí těchto informací se dozvíte, jak vkládat zprávy do fronty.

“Získávání zpráv z fronty” na stránce 744
Pomocí těchto informací získáte informace o získávání zpráv z fronty.

“Zjišťování a nastavení atributů objektu” na stránce 822
Atributy jsou vlastnosti, které definují charakteristiky objektu IBM MQ .

“Spuštění aplikací IBM MQ pomocí spouštěčů” na stránce 836
Získejte informace o spouštěčích a o tom, jak spouštět aplikace IBM MQ pomocí spouštěčů.

“Práce s rozhraním MQI a klastry” na stránce 854
Existují speciální volby pro volání a návratové kódy, které se týkají klastrování.

“Používání a psaní aplikací na systému IBM MQ for z/OS” na stránce 858
Aplikace IBM MQ for z/OS lze vytvořit z programů, které jsou spuštěny v mnoha různých prostředích. To znamená, že mohou využívat zařízení, která jsou k dispozici ve více než jednom prostředí.

“IMS a IMS přemostění aplikací na IBM MQ for z/OS” na stránce 66
Tyto informace vám pomohou psát aplikace IMS pomocí IBM MQ.

Aspekty synchronizačního bodu v aplikacích IBM MQ

Pomocí těchto informací získáte informace o používání synchronizovaných bodů v aplikacích IBM MQ .

Dvoufázové potvrzování je podporováno následujícími prostředími:

- Multi IBM MQ for Multiplatforms
- z/OS CICS Transakční server pro z/OS
- z/OS TXSeries
- z/OS IMS/ESA
- z/OS z/OS dávka s RRS
- Další externí koordinátoři používající rozhraní X/Open XA

Jednofázové potvrzování je podporováno následujícími prostředími:

- Multi IBM MQ for Multiplatforms
- z/OS z/OS dávkové

Další informace o externích rozhraních naleznete v části “Rozhraní pro externí správce synchronizačního bodu na platformě Multiplatforms” na stránce 833a v dokumentaci XA *Zpracování distribuovaných transakcí specifikace CAE: Specifikace XA*, publikované skupinou The Open Group. Správci transakcí (například CICS, IMS, Encina a Tuxedo) se mohou účastnit dvoufázového potvrzování, koordinovaného s dalšími obnovitelnými prostředky. To znamená, že funkce řazení do fronty poskytované produktem IBM MQ mohou být přeneseny do rozsahu jednotky práce spravované správcem transakcí.

Ukázky dodávané s produktem IBM MQ show IBM MQ koordinují databáze podporující standard XA. Další informace o těchto ukázkách viz “Použití ukázkových procedurálních programů IBM MQ” na stránce 1023.

V aplikaci IBM MQ můžete při každém volání put a get určit, zda má být volání pod kontrolou synchronizačního bodu. Chcete-li provést operaci vložení pod řízením synchronizačního bodu, použijte při volání MQPUT hodnotu MQPMO_SYNCPOINT v poli *Options* struktury MQPMO. Pro operaci get použijte hodnotu MQGMO_SYNCPOINT v poli *Options* struktury MQGMO. Pokud volbu explicitně nevyberete, výchozí akce závisí na platformě:

- Multi Výchozí nastavení ovládacího prvku synchronizačního bodu je NO.
- z/OS Výchozí nastavení ovládacího prvku synchronizačního bodu je YES.

Když je vydáno volání MQPUT1 s MQPMO_SYNCPOINT, výchozí chování se změní, aby byla operace vložení dokončena asynchronně. To může způsobit změnu chování některých aplikací, které spoléhají na vracená pole ve strukturách MQOD a MQMD, ale které nyní obsahují nedefinované hodnoty. Aplikace může zadat MQPMO_SYNC_RESPONSE, aby se ujistila, že operace vložení je prováděna synchronně a že jsou dokončeny všechny příslušné hodnoty pole.

Pokud vaše aplikace obdrží kód příčiny MQRC_BACKED_OUT v reakci na příkaz MQPUT nebo MQGET pod synchronizačním bodem, měla by aplikace obvykle vrátit zpět aktuální transakci pomocí příkazu MQBACK a poté, je-li to vhodné, zopakovat celou transakci. Pokud aplikace obdrží hodnotu MQRC_BACKED_OUT v reakci na volání MQCMIT nebo MQDISC, nemusí volat MQBACK.

Při každém vrácení volání MQGET se zvýší hodnota pole *BackoutCount* struktury MQMD ovlivněné zprávy. Vysoký *BackoutCount* označuje zprávu, která byla opakovaně vrácena zpět. To může označovat problém s touto zprávou, který byste měli vyšetřit. Podrobnosti o parametru *BackoutCount*viz BackoutCount .

S výjimkou dávky z/OS s RRS platí, že pokud program vydá volání MQDISC v době, kdy existují nepotvrzené požadavky, dojde k implicitní synchronizační bodu. Pokud program skončí abnormálně, dojde k implicitnímu vrácení zpět.

z/OS V systému z/OSse implicitní synchronizační bod vyskytuje také v případě, že program skončí normálně bez předchozího volání MQDISC. Program je považován za ukončený normálně, pokud TCB připojený k produktu MQ skončí normálně. Při spuštění v prostředí z/OS UNIX System Services a jazykového prostředí (LE) je pro nestandardní ukončení nebo signály vyvoláno výchozí zpracování podmínek. Obslužné rutiny podmínek LE zpracují chybový stav a TCB skončí normálně. Za těchto podmínek produkt MQ potvrdí transakci. Další informace viz Úvod do obsluhy podmínek jazykového prostředí.

z/OS V případě programů IBM MQ for z/OS můžete použít volbu MQGMO_MARK_SKIP_BACKOUT k určení, že zpráva nesmí být vrácena zpět, pokud dojde k vrácení (aby se zabránilo smyčce *MQGET-error-backout*). Informace o použití této volby viz “Přeskočení vrácení” na stránce 773.

Změny atributů fronty (buď voláním MQSET, nebo příkazy) nejsou ovlivněny potvrzením nebo zálohováním jednotek práce.

z/OS *Synchronizační body v aplikacích 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 z/OS*

V aplikaci CICS zavedete synchronizační bod pomocí příkazu EXEC CICS SYNCPOINT.

Chcete-li vrátit zpět všechny změny předchozího synchronizačního bodu, můžete použít příkaz EXEC CICS SYNCPOINT ROLLBACK. Další informace naleznete v příručce *CICS Application Programming Reference*.

Jsou-li do pracovní jednotky zapojeny další obnovitelné prostředky, správce front (ve spojení se správcem synchronizačního bodu CICS) se účastní protokolu dvoufázového potvrzování; jinak správce front provede proces jednofázového potvrzování.

Pokud aplikace CICS vydá volání MQDISC, neprovede se žádný implicitní synchronizační bod. Pokud se aplikace normálně zavře, všechny otevřené fronty se zavřou a dojde k implicitnímu potvrzení. Pokud se aplikace ukončí abnormálně, všechny otevřené fronty se zavřou a dojde k implicitnímu vrácení.

z/OS *Synchronizační body v aplikacích IMS*

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

Chcete-li vrátit zpět všechny změny od předchozího kontrolního bodu, můžete použít volání IMS ROLB (odvolání). Další informace naleznete v dokumentaci k produktu IMS .

Správce front (ve spojení se správcem synchronizačního bodu IMS) se účastní protokolu dvoufázového potvrzování, pokud jsou do transakce zapojeny i další obnovitelné prostředky.

Všechny otevřené popisovače jsou uzavřeny adaptérem IMS v synchronizačním bodu (s výjimkou prostředí BMP s dávkovým zpracováním nebo bez použití zpráv). Důvodem je skutečnost, že při provádění volání MQCONN, MQCONNX a MQOPEN může jiný uživatel zahájit další transakci a při provádění volání MQPUT nebo MQGET je provedena kontrola zabezpečení produktu IBM MQ .

Avšak v prostředí WFI (Wait-for-Input) nebo PWFI (pseudo-Wait-for-Input) IMS neoznámí IBM MQ zavření popisovačů, dokud nedorazí další zpráva nebo dokud se aplikaci nevrátí stavový kód QC. Pokud aplikace čeká v oblasti IMS a některý z těchto manipulátorů patří do spouštěných front, nedojde ke spuštění, protože jsou fronty otevřené. Z tohoto důvodu by aplikace spuštěné v prostředí WFI nebo PWFI měly explicitně MQCLOSE manipulovat s frontou před provedením GU pro IOPCB pro další zprávu.

Pokud aplikace IMS (buď BMP, nebo MPP) vydá volání MQDISC, otevřené fronty se zavřou, ale neprovede se žádný implicitní synchronizační bod. Pokud se aplikace normálně zavře, všechny otevřené fronty se

zavřou a dojde k implicitnímu potvrzení. Pokud se aplikace ukončí abnormálně, všechny otevřené fronty se zavřou a dojde k implicitnímu vrácení.

z/OS *Synchronizační body v dávkových aplikacích z/OS*

Pro dávkové aplikace můžete použít volání správy synchronizačního bodu IBM MQ : MQCMIT a MQBACK. Kvůli kompatibilitě s dřívějšími verzemi jsou CSQBCMT a CSQBBAK k dispozici jako synonyma.

Poznámka: Potřebujete-li potvrdit nebo vrátit aktualizace prostředků spravovaných různými správci prostředků, například IBM MQ a Db2, v rámci jediné pracovní jednotky, můžete použít službu RRS. Další informace viz “Správa transakcí a obnovitelné služby správce prostředků” na stránce 829.

Potvrzení změn pomocí volání MQCMIT

Jako vstup musíte zadat manipulátor připojení (*Hconn*), který je vrácen voláním MQCONN nebo MQCONNX.

Výstup z MQCMIT je kód dokončení a kód příčiny. Volání se dokončí s varováním, pokud byl synchronizační bod dokončen, ale správce front zálohoval operace vložení a získání od předchozího synchronizačního bodu.

Úspěšné dokončení volání MQCMIT označuje správci front, že aplikace dosáhla synchronizačního bodu a že všechny operace vložení a získání provedené od předchozího synchronizačního bodu byly trvalé.

Ne všechny odpovědi na selhání znamenají, že MQCMIT nebyl dokončen. Aplikace může například obdržet zprávu MQRC_CONNECTION_BROKEN.

Existuje popis volání MQCMIT v produktu MQCMIT.

Zálohování změn pomocí volání MQBACK

Jako vstup musíte zadat manipulátor připojení (*Hconn*). Použijte manipulátor vrácený 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, které byly provedeny od posledního synchronizačního bodu, byly vráceny zpět.

V části MQBACKje uveden popis volání MQBACK.

Správa transakcí a obnovitelné služby správce prostředků

Správa transakcí a obnovitelné služby správce prostředků (RRS) je prostředek z/OS , který poskytuje podporu dvoufázového synchronizačního bodu mezi zúčastněnými správci prostředků.

Aplikace může aktualizovat obnovitelné prostředky spravované různými správci prostředků z/OS , například IBM MQ a Db2, a poté tyto aktualizace potvrdit nebo vrátit zpět jako jednu pracovní jednotku. RRS poskytuje nezbytné protokolování stavu jednotky práce během normálního provádění, koordinuje zpracování synchronizačního bodu a poskytuje odpovídající informace o stavu jednotky práce během restartu subsystému.

IBM MQ for z/OS Podpora účastníků RRS umožňuje aplikacím IBM MQ v prostředí dávek, TSO a Db2 uložených procedur aktualizovat prostředky IBM MQ i jiné prostředky nežIBM MQ (například Db2). v rámci jediné logické pracovní jednotky. Informace o podpoře účastníků RRS viz *z/OS MVS Programování: Obnova prostředků*.

Aplikace IBM MQ může používat volání MQCMIT a MQBACK nebo ekvivalentní volání RRS, SRRCMIT a SRRBACK. Další informace viz “Dávkový adaptér RRS” na stránce 861.

Dostupnost služby RRS

Není-li služba RRS ve vašem systému z/OS aktivní, volání IBM MQ vydané programem propojeným se stubem RRS (CSQBRSTB nebo CSQBRRSI) vrátí hodnotu MQRC_ENVIRONMENT_ERROR.

Db2Uložené procedury

Pokud používáte uložené procedury Db2 s RRS, mějte na paměti následující:

- Uložené procedury Db2 používající službu RRS musí být spravovány správcem pracovní zátěže (WLM-managed).
- Pokud uložená procedura spravovaná pomocí Db2-managed obsahuje volání IBM MQ a je propojena se stubem RRS (CSQBRSTB nebo CSQBRRSI), vrátí volání MQCONN nebo MQCONNX chybu MQRC_ENVIRONMENT_ERROR.
- Pokud uložená procedura spravovaná WLM obsahuje volání IBM MQ a je propojena se stubem, který není RRS, volání MQCONN nebo MQCONNX vrátí hodnotu MQRC_ENVIRONMENT_ERROR, pokud se nejedná o první volání IBM MQ provedené od spuštění adresního prostoru uložené procedury.
- Pokud vaše Db2 uložená procedura obsahuje volání IBM MQ a je propojena se stubem jiného typu než RRS, prostředky IBM MQ aktualizované v této uložené proceduře nebudou potvrzeny, dokud neskončí adresní prostor uložené procedury nebo dokud následná uložená procedura neprovede operaci MQCMIT (pomocí stubu IBM MQ Batch/TSO).
- Ve stejném adresním prostoru může být souběžně provedeno více kopií stejné uložené procedury. Pokud chcete, aby produkt Db2 používal jednu kopii uložené procedury, ujistěte se, že je váš program znovu zakódován. Jinak byste mohli obdržet MQRC_HCONN_ERROR na libovolném volání IBM MQ ve vašem programu.
- Nekódujte MQCMIT ani MQBACK v uložené proceduře Db2 spravované WLM.
- Navrhněte všechny programy, které se mají spustit v jazykovém prostředí (LE).

IBM i *Synchronizační body v CICS pro aplikace IBM i*

IBM MQ for IBM i se podílí na CICS pro IBM i jednotky práce. Rozhraní MQI v rámci aplikace CICS for IBM i můžete použít k vložení a získání zpráv v rámci aktuální pracovní jednotky.

K zavedení synchronizačního bodu, který zahrnuje operace IBM MQ for IBM i , můžete použít příkaz EXEC CICS SYNCPOINT. Chcete-li vrátit všechny změny až do předchozího synchronizačního bodu, můžete použít příkaz EXEC CICS SYNCPOINT ROLLBACK.

Pokud používáte MQPUT, MQPUT1nebo MQGET s volbou MQPMO_SYNCPOINT nebo MQGMO_SYNCPOINT nastavenou v CICS pro aplikaci IBM i , nemůžete se odhlásit CICS pro IBM i , dokud IBM MQ for IBM i neodebere svou registraci jako prostředek závazku rozhraní API. Před odpojením od správce front potvrďte nebo vraťte zpět všechny nevyřízené operace vložení nebo získání. To vám umožňuje odhlásit se od CICS pro IBM i.

Multi *Synchronizační body v adresáři IBM MQ for Multiplatforms*

Podpora Syncpoint funguje na dvou typech pracovních jednotek: lokální a globální.

Lokální jednotka práce je taková, ve které jsou aktualizovány pouze prostředky správce front IBM MQ . Zde je koordinace synchronizačních bodů poskytována samotným správcem front pomocí procedury jednofázového potvrzování.

Globální jednotka práce je jednotka práce, v níž jsou aktualizovány také prostředky patřící jiným správcům prostředků, například databázím. IBM MQ může tyto jednotky práce koordinovat sama. Mohou být také koordinovány externím kontrolorem závazků. Příklad:

- Jiný správce transakcí
- IBM i Řadič vázaného zpracování IBM i

Pro úplnou integritu použijte dvoufázový potvrzovací postup. Správce transakcí a databáze podporující standard XA mohou poskytovat dvoufázové potvrzování. Příklad:

- TXSeries
- UDB
- IBM i řadič vázaného zpracování IBM i

ALW Produkty IBM MQ mohou koordinovat globální jednotky práce pomocí procesu dvoufázového potvrzování.

IBM i Produkt IBM MQ for IBM i může vystupovat jako správce prostředků pro globální pracovní jednotky v prostředí WebSphere Application Server , ale nemůže vystupovat jako správce transakcí.

Implicitní synchronizační bod

Při vkládání trvalých zpráv je produkt IBM MQ optimalizován pro vkládání trvalých zpráv do synchronizačního bodu. Více aplikací, které vkládají trvalé zprávy do stejné fronty, má lepší výkon, pokud tyto aplikace používají synchronizační bod. Důvodem je menší soupeření o frontu, pokud je k vložení trvalých zpráv použit synchronizační bod.

Produkt **ImplSyncOpenOutput** přidá implicitní synchronizační bod, když aplikace vkládají trvalé zprávy mimo synchronizační bod. To poskytuje zlepšení výkonu, aniž by aplikace byly informovány o implicitním synchronizačním bodu.

Implicitní synchronizační bod poskytuje zvýšení výkonu pouze v případě, že do fronty vkládá více aplikací, protože snižuje soupeření o frontu. Parametr **ImplSyncOpenOutput** tedy určuje minimální počet aplikací, které mají před přidáním implicitního synchronizačního bodu otevřenou frontu pro výstup. Výchozí hodnota je 2. To znamená, že pokud neurčíte **ImplSyncOpenOutput**, bude implicitní synchronizační bod přidán pouze v případě, že do fronty vkládáte více aplikací.

Další informace viz Ladění parametrů .

Místní jednotky práce na platformě Multiplatforms
Pracovní jednotky, které zahrnují pouze správce front, se nazývají *lokální* pracovní jednotky. Koordinaci synchronizačních bodů zajišťuje sám správce front (interní koordinace) pomocí procesu jednofázového potvrzování.

Chcete-li spustit lokální pracovní jednotku, aplikace vydá požadavky MQGET, MQPUT nebo MQPUT1 s uvedením příslušné volby synchronizačního bodu. Transakce je potvrzena pomocí MQCMIT nebo odvolána pomocí MQBACK. Pracovní jednotka však skončí i v případě, že dojde k úmyslnému nebo neúmyslnému přerušení připojení mezi aplikací a správcem front.

Pokud se aplikace odpojí (MQDISC) od správce front, zatímco globální jednotka práce koordinovaná produktem IBM MQ je stále aktivní, dojde k pokusu o potvrzení transakce. Pokud se však aplikace ukončí bez odpojení, jednotka práce se odvolá, protože se má za to, že aplikace byla ukončena nestandardně.

Globální jednotky práce na platformě Multiplatforms
Globální pracovní jednotky používejte v případě, že potřebujete také zahrnout aktualizace prostředků náležejících jiným správcům prostředků.

Zde může být koordinace interní nebo externí pro správce front:

Interní koordinace synchronizačních bodů

Koordinace správce front globálních pracovních jednotek není produktem IBM MQ for IBM i ani IBM MQ for z/OSpodporována. Není podporován v IBM MQ MQI client prostředí.

Zde IBM MQ provádí koordinaci. Chcete-li spustit globální jednotku práce, aplikace vydá volání MQBEGIN.

Jako vstup pro volání MQBEGIN musíte zadat manipulátor připojení (*Hconn*), který je vrácen voláním MQCONN nebo MQCONNX. Tento manipulátor představuje připojení ke správci front IBM MQ .

Aplikace zadá požadavky MQGET, MQPUT nebo MQPUT1 s určením příslušné volby synchronizačního bodu. To znamená, že můžete použít příkaz MQBEGIN k zahájení globální pracovní jednotky, která aktualizuje lokální prostředky, prostředky patřící jiným správcům prostředků nebo obojí. Aktualizace prostředků náležejících jiným správcům prostředků se provádějí pomocí rozhraní API tohoto správce prostředků. Rozhraní MQI však nelze použít k aktualizaci front, které patří jiným správcům front.
Před spuštěním dalších pracovních jednotek (lokálních nebo globálních) zadejte příkaz MQCMIT nebo MQBACK.

Globální jednotka práce je potvrzena pomocí MQCMIT. Tato operace zahájí dvoufázové potvrzování všech správců prostředků zapojených do jednotky práce. Používá se proces dvoufázového potvrzování, při kterém jsou nejprve všichni správci prostředků (například správci databází kompatibilní se standardem XA, jako např. Db2, Oraclea Sybase) požádáni o přípravu na potvrzení. Pouze v případě, že jsou všechny připraveny, jsou požádány o potvrzení. Pokud některý správce prostředků signalizuje, že nemůže provést potvrzení, bude místo toho vyzván k vrácení. Případně můžete pomocí příkazu MQBACK odvolat aktualizace všech správců prostředků.

Pokud se aplikace odpojí (MQDISC), zatímco globální jednotka práce je stále aktivní, jednotka práce je potvrzena. Pokud se však aplikace ukončí bez odpojení, jednotka práce se odvolá, protože se má za to, že aplikace byla ukončena nestandardně.

Výstup příkazu MQBEGIN je kód dokončení a kód příčiny.

Při použití příkazu MQBEGIN ke spuštění globální pracovní jednotky jsou zahrnuti všichni externí správci prostředků, kteří byli konfigurováni se správcem front. Avšak volání spustí pracovní jednotku, ale dokončí se s varováním, pokud:

- Neexistují žádní zúčastnění správci prostředků (tj. s tímto správcem front nebyli konfigurováni žádní správci prostředků).

, nebo

- Jeden nebo více správců prostředků není k dispozici.

V těchto případech musí jednotka práce obsahovat aktualizace pouze těch správců prostředků, kteří byli k dispozici při spuštění jednotky práce.

Pokud některý ze správců prostředků nemůže potvrdit své aktualizace, budou všichni správci prostředků instruováni, aby své aktualizace odvolali, a produkt MQCMIT bude dokončen s varováním. Za neobvyklých okolností (obvykle zásah operátora) může volání MQCMIT selhat, pokud někteří správci prostředků své aktualizace potvrdí, ale jiní je odvolají; práce je považována za dokončenou s *smíšeným* výsledkem. Tyto výskyty jsou diagnostikovány v protokolu chyb správce front, aby bylo možné provést nápravnou akci.

Operace MQCMIT globální pracovní jednotky je úspěšná, pokud všichni správci prostředků, kteří se na ní podíleli, potvrdí své aktualizace.

Popis volání MQBEGIN viz MQBEGIN.

Externí koordinace synchronizačních bodů

K tomu dochází v případě, že byl vybrán jiný koordinátor synchronizačního bodu než IBM MQ , například CICS, Encina nebo Tuxedo.

V této situaci systémy IBM MQ for AIX, Linux, and Windows registrují svůj zájem o výsledek transakce s koordinátorem synchronizačního bodu, aby mohly podle potřeby potvrdit nebo odvolat nepotvrzené operace get nebo put. Koordinátor externího synchronizačního bodu určuje, zda jsou poskytnuty jednofázové nebo dvoufázové protokoly vázaného zpracování.

Při použití externího koordinátora nelze zadat příkazy MQCMIT, MQBACK a MQBEGIN. Volání těchto funkcí se nezdaří s kódem příčiny MQRC_ENVIRONMENT_ERROR.

Způsob spuštění externě koordinované pracovní jednotky závisí na programovacím rozhraní poskytovaném koordinátorem synchronizačního bodu. Může být vyžadováno explicitní volání. Je-li vyžadováno explicitní volání a zadáte-li volání MQPUT s volbou MQPMO_SYNCPOINT, když není spuštěna jednotka práce, vrátí se kód dokončení MQRC_SYNCPOINT_NOT_AVAILABLE.

Rozsah pracovní jednotky je určen koordinátorem synchronizačního bodu. Stav připojení mezi aplikací a správcem front má vliv na úspěch nebo selhání volání MQI, která aplikace vydá, nikoli na stav pracovní jednotky. Aplikace může například během aktivní pracovní jednotky odpojit a znovu se připojit ke správci front a provádět další operace MQGET a MQPUT ve stejné pracovní jednotce. Toto je známé jako nevyřízené odpojení.

Můžete použít volání IBM MQ API v programech CICS , ať už se rozhodnete použít schopnosti XA CICS. Pokud nepoužíváte prostředí XA, nebudou operace vložení a získání zpráv do front a z front spravovány

v rámci CICS atomických pracovních jednotek. Jedním z důvodů pro výběr této metody je, že celková konzistence pracovní jednotky pro vás není důležitá.

Pokud je pro vás důležitá integrita vašich pracovních jednotek, musíte použít XA. Při použití XA produkt CICS používá protokol dvoufázového potvrzování, který zajišťuje, že všechny prostředky v rámci pracovní jednotky jsou aktualizovány společně.

Další informace o nastavení podpory transakcí viz Scénáře podpory transakcía také dokumentace k produktu TXSeries CICS , například *TXSeries for Multiplatforms CICS Administration Guide for Open Systems*.

Multi *Implicitní synchronizační bod na multiplatformách*

Implicitní podpora synchronizačního bodu umožňuje vložení trvalých zpráv mimo synchronizační bod.

Při vkládání trvalých zpráv je produkt IBM MQ optimalizován pro vkládání trvalých zpráv do synchronizačního bodu. Více aplikací souběžně vkládajících trvalé zprávy do stejné fronty obvykle provádí lepší výkon, pokud tyto aplikace používají synchronizační bod. Je to proto, že strategie zamykání produktu IBM MQ je efektivnější, pokud se při vkládání trvalých zpráv používá synchronizační bod.

Parametr **ImplSyncOpenOutput** v souboru `qm.ini` určuje, zda lze implicitní synchronizační bod přidat, když aplikace vkládají trvalé zprávy mimo synchronizační bod. To může zajistit zlepšení výkonu, aniž by si aplikace byly vědomy implicitního synchronizačního bodu.

Implicitní synchronizační bod poskytuje zvýšení výkonu pouze v případě, že do fronty vkládá více aplikací současně, protože snižuje soupeření o zámek. **ImplSyncOpenOutput** uvádí minimální počet aplikací, které mají otevřenou frontu pro výstup, než lze přidat implicitní synchronizační bod. Výchozí hodnota je *2*. To znamená, že pokud explicitně neurčíte **ImplSyncOpenOutput**, bude implicitní synchronizační bod přidán pouze v případě, že do fronty vkládá více aplikací.

Přidáte-li implicitní synchronizační bod, statistika to odráží a může se zobrazit výstup transakce z produktu **runmqsc display conn**.

Nastavte **ImplSyncOpenOutput**=*OFF* , pokud nechcete přidat implicitní synchronizační bod.

Další informace viz Ladění parametrů .

Rozhraní pro externí správce synchronizačního bodu na platformě Multiplatforms

Produkt IBM MQ for Multiplatforms podporuje koordinaci transakcí externími správci synchronizačních bodů, kteří používají rozhraní XA sdružení X/Open.

Někteří správci transakcí XA (TXSeries) vyžadují, aby každý správce prostředků XA dodala svůj název. Jedná se o řetězec s názvem `name` ve struktuře přepínače XA.

- ALW Správce prostředků pro systém IBM MQ v systému AIX, Linux, and Windows má název MQSeries_XA_RMI.
- IBM i Pro systém IBM ije název správce prostředků MQSeries XA RMI.

Další podrobnosti o rozhraních XA naleznete v dokumentaci XA *CAE Specification Distributed Transaction Processing: Specifikace XA*, publikované společností The Open Group.

V konfiguraci XA produkt IBM MQ for Multiplatforms plní roli správce prostředků XA. Koordinátor synchronizačního bodu XA může spravovat sadu správců prostředků XA a synchronizovat potvrzení nebo vrácení transakcí v obou správcích prostředků. Takto to funguje pro staticky registrovaného správce prostředků:

1. Aplikace oznámí koordinátorovi synchronizačního bodu, že chce spustit transakci.
2. Koordinátor synchronizačního bodu vydá volání všem správcům prostředků, které zná, aby je upozornil na aktuální transakci.
3. Aplikace vydává volání pro aktualizaci prostředků spravovaných správci prostředků přidruženými k aktuální transakci.

4. Aplikace požaduje, aby koordinátor synchronizačního bodu transakci buď potvrdil, nebo odvolala.
5. Koordinátor synchronizačního bodu vydá volání pro každého správce prostředků s použitím protokolů dvoufázového potvrzování k dokončení transakce podle požadavku.

Specifikace XA vyžaduje, aby každý správce prostředků poskytoval strukturu s názvem Přepínač XA. Tato struktura deklaruje schopnosti správce prostředků a funkce, které má volat koordinátor synchronizačního bodu.

Existují dvě verze této struktury:

Tabulka 128. Verze přepínače XA

Verze	**Popis**
MQRMIXASwitch	Statická správa prostředků XA
MQRMIXASwitchDynamic	Dynamická správa prostředků XA

Seznam knihoven obsahujících tuto strukturu viz IBM MQ Struktura přepínače XA.

Metoda, která musí být použita k jejich propojení s koordinátorem synchronizačního bodu XA, je definována koordinátorem. Prostudujte dokumentaci poskytnutou tímto koordinátorem a určete, jak povolit produktu IBM MQ spolupracovat s koordinátorem synchronizačního bodu XA.

Struktura *xa_info* předaná koordinátorem synchronizačního bodu v libovolném volání *xa_open* může být názvem správce front, který má být spravován. Má stejnou formu jako název správce front předaný MQCONN nebo MQCONNX a může být prázdný, pokud má být použit výchozí správce front. Můžete však použít dva další parametry TPM a AXLIB

Produkt TPM vám umožňuje zadat IBM MQ název správce transakcí, například CICS. AXLIB umožňuje určit skutečný název knihovny ve správci transakcí, kde 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 viz Parametry CHANNEL, TRPTYPE, CONNAME a QMNAME řetězce xa_open.

Omezení

1. Globální jednotky práce nejsou povoleny se sdíleným Hconn (jak je popsáno v části “Sdílená (nezávislá na vláknu) připojení s MQCONNX” na stránce 716.
2. IBM i Produkt IBM MQ for IBM i nepodporuje dynamickou registraci správců prostředků XA.

   Jediný podporovaný správce transakcí je WebSphere Application Server.
3. Windows 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 k jednotlivým správcům front efektivní připojení, a proto se na něj vztahuje pravidlo, že v daném okamžiku je povoleno pouze jedno připojení.

   Poznámka: Poznámka: Aplikace klienta JMS (KLIENT JEE) spuštěná na serveru JEE toto omezení nemá, takže jedna transakce spravovaná serverem JEE může koordinovat více správců front ve stejné transakci. Aplikace serveru JMS spuštěná v režimu vazeb však stále podléhá pravidlu, že v daném okamžiku je povoleno pouze jedno připojení.
5. Všechny aplikace spuštěné s použitím koordinátora synchronizačního bodu se mohou připojit pouze ke správci front spravovanému koordinátorem, protože jsou již efektivně připojeny k tomuto správci front. Musí zadat příkaz MQCONN nebo MQCONNX, aby získali manipulátor připojení, a před ukončením musí zadat příkaz MQDISC. Případně mohou použít uživatelskou proceduru UE014015 pro TXSeries CICS.

IBM i *Rozhraní pro externího správce synchronizačního bodu IBM i*

IBM MQ for IBM i může používat nativní IBM i vázané zpracování jako externí koordinátor synchronizačního bodu.

Připojení nezávislá na vláknu (sdílená) nejsou povolena s vázaným zpracováním. Další informace o možnostech vázaného zpracování produktu IBM inaleznete v příručce *IBM i Programming: Backup and Recovery Guide, SC21-8079* .

Chcete-li spustit prostředky vázaného zpracování IBM i , použijte systémový příkaz STRCMTCTL. Chcete-li ukončit vázané zpracování, použijte systémový příkaz ENDCMTCTL.

Poznámka: Výchozí hodnota *Rozsah definice závazku* je *ACTGRP. Toto musí být definováno jako *JOB pro IBM MQ pro IBM i. Příklad:

```
STRCMTCTL LCKLVL(*ALL) CMTSCOPE(*JOB)
```

Produkt IBM MQ for IBM i může také provádět lokální pracovní jednotky obsahující pouze aktualizace prostředků IBM MQ . Volba mezi lokálními pracovními jednotkami a účastí v globálních pracovních jednotkách koordinovaných produktem IBM i se provádí v každé aplikaci, když aplikace volá MQPUT, MQPUT1nebo MQGET se zadáním MQPMO_SYNCPOINT nebo MQGMO_SYNCPOINT nebo MQBEGIN. Pokud není vázané zpracování aktivní, když je vydáno první takové volání, produkt IBM MQ spustí lokální pracovní jednotku a všechny další pracovní jednotky pro toto připojení k produktu IBM MQ také použijí lokální pracovní jednotky, bez ohledu na to, zda je vázané zpracování spuštěno. Chcete-li potvrdit lokální pracovní jednotku, použijte MQCMIT. Chcete-li vrátit zpět lokální pracovní jednotku, použijte příkaz MQBACK. Volání operace commit a rollback systému IBM i , jako např. příkaz CL COMMIT, nemají žádný vliv na IBM MQ lokální pracovní jednotky.

Chcete-li použít IBM MQ for IBM i s nativním IBM i vázaným zpracováním jako externí koordinátor synchronizačních bodů, ujistěte se, že každá úloha s vázaným zpracováním je aktivní a že používáte IBM MQ v úloze s jedním vláknem. Pokud voláte MQPUT, MQPUT1nebo MQGET s uvedením MQPMO_SYNCPOINT nebo MQGMO_SYNCPOINT v úloze s více podprocesy, ve které bylo spuštěno vázané zpracování, volání selže s kódem příčiny MQRC_SYNCPOINT_NOT_AVAILABLE.

Je možné použít lokální pracovní jednotky a volání MQCMIT a MQBACK ve vícevláknové úloze.

Pokud voláte MQPUT, MQPUT1nebo MQGET a uvedete MQPMO_SYNCPOINT nebo MQGMO_SYNCPOINT po spuštění vázaného zpracování, produkt IBM MQ for IBM i se přidá jako prostředek vázaného zpracování rozhraní API do definice vázaného zpracování. Toto je obvykle první takové volání v úloze. I když existují nějaké prostředky vázaného zpracování rozhraní API registrované pod konkrétní definicí vázaného zpracování, nemůžete ukončit vázané zpracování pro tuto definici.

Produkt IBM MQ for IBM i odebere svou registraci jako prostředek vázaného zpracování rozhraní API při odpojení od správce front, pokud v aktuální transakci neexistují žádné nevyřízené operace MQI.

Pokud se odpojíte od správce front v době, kdy existují nevyřízené operace MQPUT, MQPUT1nebo MQGET v aktuální transakci, produkt IBM MQ for IBM i zůstane registrován jako prostředek vázaného zpracování rozhraní API, aby byl upozorněn na další potvrzení nebo odvolání. Po dosažení dalšího synchronizačního bodu produkt IBM MQ for IBM i potvrdí nebo odvolá změny podle potřeby. Aplikace se může odpojit a znovu připojit ke správci front během aktivní pracovní jednotky a provádět další operace MQGET a MQPUT v rámci stejné pracovní jednotky (jedná se o nevyřízené odpojení).

Pokud se pokusíte vydat systémový příkaz ENDCMTCTL pro tuto definici vázaného zpracování, vydá se zpráva CPF8355 , která označuje, že nevyřízené změny byly aktivní. Tato zpráva se také objeví v protokolu úlohy po ukončení úlohy. Chcete-li se tomu vyhnout, potvrďte nebo odvolejte všechny nevyřízené operace IBM MQ for IBM i a odpojte se od správce front. Proto použití příkazů COMMIT nebo ROLLBACK před ENDCMTCTL umožňuje úspěšné dokončení koncového vázaného zpracování.

Používáte-li řízení vázaného zpracování IBM i jako externí koordinátor synchronizačního bodu, nemůžete zadat volání MQCMIT, MQBACK a MQBEGIN. Volání těchto funkcí se nezdaří s kódem příčiny MQRC_ENVIRONMENT_ERROR.

Chcete-li potvrdit nebo odvolat (tj. odvolat) svou pracovní jednotku, použijte jeden z programovacích jazyků, který podporuje vázané zpracování. Příklad:

- Příkazy CL: COMMIT a ROLLBACK
- Funkce programování v C ILE: _Rcommit a _Rrollback
- ILE RPG: COMMIT a ROLBK
- COBOL/400: COMMIT a ROLLBACK

Použijete-li řízení vázaného zpracování IBM i jako externí koordinátor synchronizačních bodů s produktem IBM MQ for IBM i, produkt IBM i provede protokol dvoufázového potvrzování, kterého se produkt IBM MQ účastní. Vzhledem k tomu, že každá pracovní jednotka je potvrzena ve dvou fázích, může se stát, že správce front nebude k dispozici pro druhou fázi poté, co byl v první fázi odhlasován pro potvrzení. K tomu může dojít například v případě, že jsou ukončeny interní úlohy správce front. V této situaci protokol úlohy provádějící potvrzení obsahuje zprávu CPF835F , která označuje, že operace potvrzení nebo odvolání selhala. Předchozí zprávy označují příčinu problému, zda se vyskytl během operace potvrzení nebo odvolání, a také ID logické pracovní jednotky (LUWID) pro nezdařenou pracovní jednotku.

Pokud byl problém způsoben selháním prostředku vázaného zpracování rozhraní API IBM MQ během potvrzení nebo odvolání připravené pracovní jednotky, můžete použít příkaz WRKMQMTRN k dokončení operace a obnovení integrity transakce. Příkaz vyžaduje, abyste znali LUWID pracovní jednotky pro potvrzení a vrácení zpět.

Spuštění aplikací IBM MQ pomocí spouštěčů

Získejte informace o spouštěčích a o tom, jak spouštět aplikace IBM MQ pomocí spouštěčů.

Některé aplikace systému IBM MQ , které obsluhují fronty, běží nepřetržitě, takže jsou vždy k dispozici pro načtení zpráv, které přicházejí do front. To však nemusí být žádoucí, pokud je počet zpráv přicházejících do front nepředvídatelný. V tomto případě mohou aplikace spotřebovávat systémové prostředky i v případě, že nejsou k dispozici žádné zprávy k načtení.

Produkt IBM MQ poskytuje prostředek, který umožňuje automatické spuštění aplikace, když jsou k dispozici zprávy k načtení. Toto zařízení je známé jako *spouštění*.

Informace o spouštění kanálů viz Spouštěcí kanály.

Co je spouštění?

Správce front definuje určité podmínky jako *spouštěcí události*.

Pokud je pro frontu povoleno spouštění a dojde k události spouštěče, správce front odešle *zprávu spouštěče* do fronty s názvem *inicializační fronta*. Přítomnost zprávy spouštěče v inicializační frontě označuje, že došlo k události spouštěče.

Zprávy spouštěče generované správcem front nejsou trvalé. To snižuje protokolování (což vede ke zlepšení výkonu) a minimalizaci duplicit během restartu, takže se zkvalitňuje doba restartu.

Program, který zpracovává inicializační frontu, se nazývá *aplikace monitoru spouštěčů*a jeho funkcí je číst zprávu spouštěče a provádět odpovídající akce na základě informací obsažených ve zprávě spouštěče. Tato akce obvykle slouží ke spuštění jiné aplikace pro zpracování fronty, která vygenerovala zprávu spouštěče. Z pohledu správce front není na aplikaci monitoru spouštěčů nic zvláštního; je to prostě jiná aplikace, která čte zprávy z fronty (inicializační fronta).

Je-li pro frontu povoleno spouštění, můžete k ní vytvořit *objekt definice procesu* . Tento objekt obsahuje informace o aplikaci, která zpracovává zprávu, která způsobila událost spouštěče. Je-li vytvořen objekt definice procesu, správce front tyto informace extrahuje a umístí je do zprávy spouštěče pro použití aplikací monitoru spouštěčů. Název definice procesu přidružené k frontě je dán atributem lokální fronty `ProcessName` . Každá fronta může určovat jinou definici procesu a několik front může sdílet stejnou.

Chcete-li spustit spuštění kanálu, nemusíte definovat objekt definice procesu. Místo toho se použije definice přenosové fronty.

Spouštění je podporováno klienty systému IBM MQ spuštěnými na systému AIX, Linux, and Windows. Aplikace spuštěná v klientském prostředí je stejná jako aplikace spuštěná v úplném prostředí IBM MQ , kromě toho, že ji propojíte s klientskými knihovnami. Avšak monitor spouštěčů a aplikace, která se má spustit, musí být ve stejném prostředí.

Spouštění zahrnuje:

Fronta aplikací
Fronta aplikací je lokální fronta, která v případě, že má nastaveno spouštění a jsou splněny podmínky, vyžaduje zápis zpráv spouštěče.

Definice procesu
Fronta aplikací může mít přidružený *objekt definice procesu* , který uchovává podrobnosti o aplikaci, která získá zprávy z fronty aplikace. (Seznam atributů naleznete v části Atributy pro definice procesů .)

Nezapomeňte, že pokud chcete, aby spouštěč spustil kanál, nemusíte definovat objekt definice procesu.

Přenosová fronta
Pokud chcete, aby spouštěč spustil kanál, potřebujete přenosovou frontu.

Pro přenosovou frontu na jakékoli jiné platformě než Linuxmůže atribut *TriggerData* přenosové fronty uvádět název kanálu, který se má spustit. To může nahradit definici procesu pro spouštění kanálů, ale používá se pouze v případě, že není vytvořena definice procesu.

událost spouštěče
Spouštěč událostí je událost, která způsobí, že správce front vygeneruje zprávu spouštěče. Obvykle se jedná o zprávu přicházející do fronty aplikací, ale může se vyskytnout i jindy. Viz například “Podmínky pro událost spouštěče” na stránce 842.

Produkt IBM MQ má řadu voleb, které vám umožňují řídit podmínky, které způsobují událost spouštěče (viz “Řízení událostí spouštěče” na stránce 846).

zpráva spouštěče
Správce front vytvoří *zprávu spouštěče* , když rozpozná událost spouštěče. Zkopíruje do zprávy spouštěče informace o aplikaci, která se má spustit. Tyto informace pocházejí z fronty aplikací a z objektu definice procesu přidruženého k frontě aplikace.

Zprávy spouštěče mají pevný formát (viz “Formát zpráv spouštěče” na stránce 853).

Inicializační fronta
Inicializační fronta je lokální fronta, do které 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 každá z nich je přidružena k jedné nebo více frontám aplikací.

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 nepřetržitě 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. Vydá příkaz ke spuštění aplikace, který načte zprávy přicházející do fronty aplikace a předá jí informace obsažené v záhlaví zprávy spouštěče, které obsahuje název fronty aplikace.

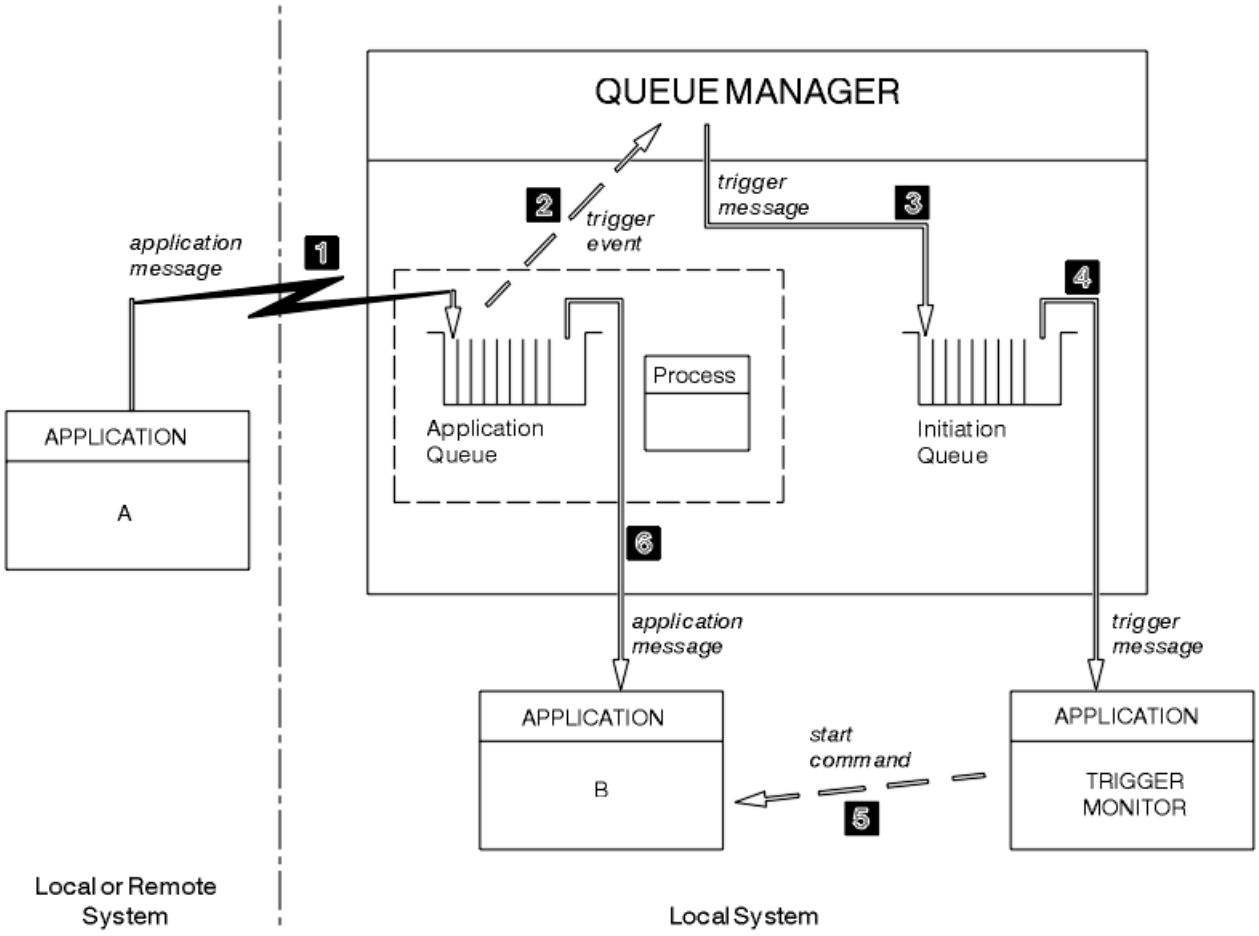
Na všech platformách je za spouštění kanálů zodpovědný speciální monitor spouštěčů známý jako inicializátor kanálu.

z/OS V systému z/OSje inicializátor kanálu obvykle spuštěn ručně nebo jej lze provést automaticky při spuštění správce front změnou hodnoty CSQINP2 ve spouštěcím kódu JCL správce front.

Multi V systému Multiplatformsse iniciátor kanálu spustí automaticky při spuštění správce front nebo jej lze spustit ručně pomocí příkazu **runmqchi** .

Další informace viz téma “Zpracování inicializační fronty monitory spouštěčů” na stránce 849.

Chcete-li pochopit, jak spouštění funguje, zvažte Obrázek 95 na stránce 838, což je příklad spouštěče typu FIRST (MQTT_FIRST).



Obrázek 95. Tok zpráv aplikace a spouštěče

V produktu Obrázek 95 na stránce 838je posloupnost událostí:

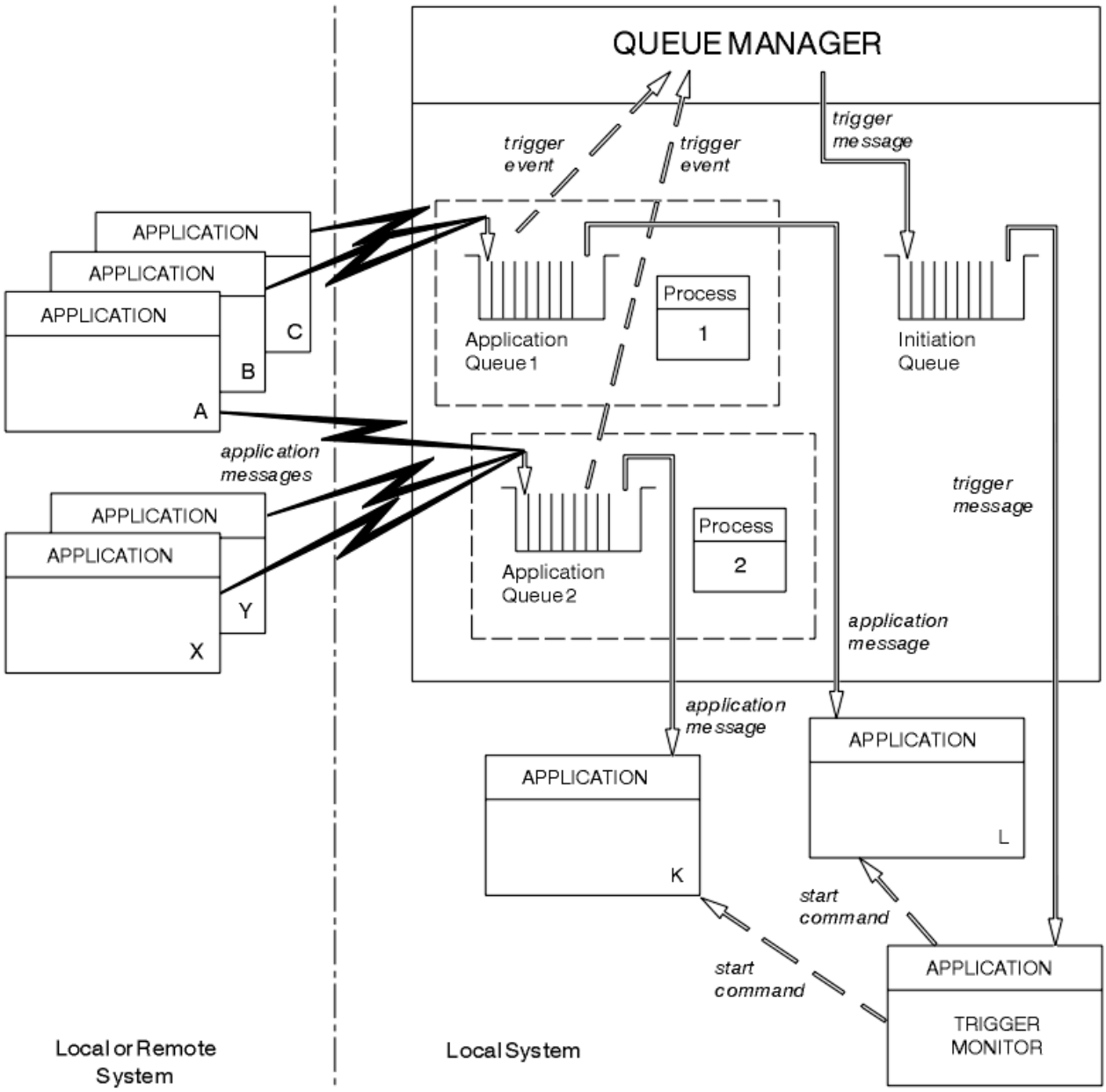
1. Aplikace A, která může být buď lokální, nebo vzdálená pro správce front, vloží zprávu do fronty aplikací. Žádná aplikace nemá tuto frontu otevřenou pro vstup. Tato skutečnost je však relevantní pouze pro typ spouštěče FIRST a DEPTH.
2. Správce front kontroluje, zda jsou splněny podmínky, za kterých musí vygenerovat událost spouštěče. Jsou a je generována událost spouštěče. Informace uchovávané v přidruženém objektu definice procesu se použijí při vytváření zprávy spouštěče.
3. Správce front vytvoří zprávu spouštěče a vloží ji do inicializační fronty přidružené k této frontě aplikace, ale pouze v případě, že aplikace (monitor spouštěčů) má inicializační frontu otevřenou pro vstup.
4. Monitor spouštěčů načte zprávu spouštěče z inicializační fronty.
5. Monitor spouštěčů vydá příkaz ke spuštění aplikace B (serverová aplikace).
6. Aplikace B otevře frontu aplikací a načte zprávu.

Poznámka:

1. Pokud je 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. Pokud inicializační fronta není otevřena pro vstup, správce front negeneruje žádné zprávy spouštěče; čeká, dokud aplikace neotevře inicializační frontu pro vstup.
3. Při použití spouštěče pro kanály použijte typ spouštěče FIRST nebo DEPTH.

4. Spuštěné aplikace se spouštějí pod ID uživatele a skupinou uživatele, který spustil monitor spouštěčů, pod uživatelem CICS nebo pod uživatelem, který spustil správce front.

Dosud byl vztah mezi frontami v rámci spouštěče pouze jeden ku jednomu. Zvažte možnost Obrázek 96 na stránce 839.



Obrázek 96. Relace front v rámci spouštěče

Fronta aplikací má k sobě přidružený objekt definice procesu, který uchovává podrobnosti o aplikaci, která bude zprávu zpracovávat. Správce front umístí informace do zprávy spouštěče, takže je nutná pouze jedna inicializační fronta. Monitor spouštěčů extrahuje tyto informace ze zprávy spouštěče a spustí příslušnou aplikaci pro zpracování zprávy v každé frontě aplikací.

Nezapomeňte, že pokud chcete spustit spuštění kanálu, nemusíte definovat objekt definice procesu. Definice přenosové fronty může určit kanál, který má být spuštěn.

Další informace o spouštění aplikací IBM MQ pomocí spouštěčů naleznete v následujících odkazech:

- “Předpoklady pro spuštění” na stránce 840
- “Podmínky pro událost spouštěče” na stránce 842

- “Řízení událostí spouštěče” na stránce 846
- “Návrh aplikace, která používá spuštěné fronty” na stránce 848
- “Zpracování inicializační fronty monitory spouštěčů” na stránce 849
- “Vlastnosti zpráv spouštěče” na stránce 852
- “Když spuštění nefunguje” na stránce 854

Související pojmy

“Přehled rozhraní fronty zpráv” na stránce 700
Informace o komponentách rozhraní MQI (Message Queue Interface).

“Připojení ke správci front a odpojení od něj” na stránce 713
Chcete-li používat programovací služby IBM MQ , musí mít program připojení ke správci front. Pomocí těchto informací se dozvíte, jak se připojit ke správci front a jak se od něj odpojit.

“Otevírání a zavírání objektů” na stránce 720
Tyto informace poskytují náhled na otevírání a zavírání objektů IBM MQ .

“Vkládání zpráv do fronty” na stránce 730
Pomocí těchto informací se dozvíte, jak vkládat zprávy do fronty.

“Získávání zpráv z fronty” na stránce 744
Pomocí těchto informací získáte informace o získávání zpráv z fronty.

“Zjišťování a nastavení atributů objektu” na stránce 822
Atributy jsou vlastnosti, které definují charakteristiky objektu IBM MQ .

“Potvrzení a zálohování jednotek práce” na stránce 825
Tyto informace popisují, jak potvrdit a vrátit zpět všechny obnovitelné operace get a put, které se vyskytly v transakci.

“Práce s rozhraním MQI a klastry” na stránce 854
Existují speciální volby pro volání a návratové kódy, které se týkají klastrování.

“Používání a psaní aplikací na systému IBM MQ for z/OS” na stránce 858
Aplikace IBM MQ for z/OS lze vytvořit z programů, které jsou spuštěny v mnoha různých prostředích. To znamená, že mohou využívat zařízení, která jsou k dispozici ve více než jednom prostředí.

“IMS a IMS přemostění aplikací na IBM MQ for z/OS” na stránce 66
Tyto informace vám pomohou psát aplikace IMS pomocí IBM MQ.

Předpoklady pro spuštění

Pomocí těchto informací získáte informace o krocích, které je třeba provést před použitím spouštěče.

Před tím, než vaše aplikace využije výhody spuštění, proveďte následující kroky:

1. Proveďte jednu z následujících akcí:
   a. Vytvořte inicializační frontu pro vaši 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 kterou může vaše aplikace používat (obvykle se jedná o název SYSTEM.DEFAULT.INITIATION.QUEUE nebo, pokud spouštíte kanály se spouštěči, SYSTEM.CHANNEL.INITQ) a zadejte jeho název do pole *InitiationQName* fronty aplikací.

2. Přidružte inicializační frontu k frontě aplikace. Správce front může vlastnit více než jednu inicializační frontu. Možná budete chtít, aby některé z vašich front aplikací byly obsluhovány různými programy.

V takovém případě můžete použít jednu inicializační frontu pro každý obsluhující program, i když to není nutné. Zde je příklad, jak vytvořit frontu aplikací:

```
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 programu CL 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 spouštějete aplikaci, vytvořte objekt definice procesu, který bude obsahovat informace související s aplikací, která má obsluhovat vaši frontu aplikací. Chcete-li například spustit mzdovou transakci CICS s názvem PAYR, postupujte takto:

```
DEFINE PROCESS (process.name) +
       REPLACE +
       DESCR ('process description') +
       APPLICID ('PAYR') +
       APPLTYPE (CICS) +
       USERDATA ('Payroll data')
```

IBM i Zde je výpis z programu CL 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, zkopíruje informace z atributů objektu definice procesu do zprávy spouštěče.

Platforma	Vytvoření objektu definice procesu
Systémy AIX, Linux, and Windows	Použijte DEFINE PROCESS nebo SYSTEM.DEFAULT.PROCESS a úpravy pomocí příkazu ALTER PROCESS
z/OS z/OS z/OS	Použijte DEFINE PROCESS (viz vzorový kód v kroku “3” na stránce 841) nebo použijte operace a ovládací panely.
IBM i IBM i IBM i	Použijte CL program obsahující kód jako v kroku “3” na stránce 841.

4. Volitelné: Vytvořte definici přenosové fronty a použijte mezery pro atribut **ProcessName** .

Atribut **TrigData** může obsahovat název kanálu, který má být spuštěn, nebo může zůstat prázdný. Je-li s výjimkou systému IBM MQ for z/OSponecháno prázdné, hledá iniciátor kanálu soubory definice kanálu, 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 pro určení vlastností aplikace, která má obsluhovat frontu aplikace, přidružte objekt procesu k frontě aplikace tak, že jej pojmenujete v atributu **ProcessName** fronty.

Platforma	Použít příkazy
Systémy AIX, Linux, and Windows	ALTER QLOCAL
z/OS z/OS z/OS	ALTER QLOCAL
IBM i IBM i IBM i	CHGMQMQ

6. Spusťte instance monitorů spouštěčů IBM i (nebo serverů spouštěčů v IBM MQ for IBM i) , které mají obsluhovat vámi definované inicializační fronty. Další informace viz “Zpracování inicializační fronty monitory spouštěčů” na stránce 849.

Chcete-li si být vědomi všech nedoručených zpráv spouštěče, ujistěte se, že váš správce front má definovanou frontu nedoručených zpráv (nedoručených zpráv). Do pole správce front *DeadLetterQName* zadejte název fronty.

Poté můžete nastavit požadované podmínky spouštěče pomocí atributů objektu fronty, který definuje vaši frontu aplikace. Další informace viz “Řízení událostí spouštěče” na stránce 846.

Podmínky pro událost spouštěče

Správce front vytvoří zprávu spouštěče při splnění podmínek uvedených v tomto tématu.

Odkazy na sdílené fronty v tomto tématu znamenají sdílené fronty ve skupině sdílení front, které jsou k dispozici pouze v systému IBM MQ for z/OS.

Následující podmínky způsobí, že správce front vytvoří zprávu spouštěče:

1. Zpráva je *vložena* do fronty.
2. Zpráva má prioritu větší nebo rovnou prahové prioritě spouštěče fronty. Tato priorita je nastavena v atributu lokální fronty **TriggerMsgPriority** ; pokud je nastavena na nulu, všechny zprávy se kvalifikují.
3. Počet zpráv ve frontě s prioritou větší 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. V případě nesdílených lokálních front správce front počítá potvrzenou i nepotvrzenou zprávu při posuzování, zda existují podmínky pro událost spouštěče. V důsledku toho může být aplikace spuštěna, když pro ni neexistují žádné zprávy, které by bylo možné načíst, protože zprávy ve frontě nebyly potvrzeny. V této situaci zvažte použití volby čekání s vhodným parametrem *WaitInterval*, aby aplikace čekala na příchod svých zpráv.

   b. Pro lokální sdílené fronty počítá správce front pouze potvrzené zprávy.

4. Pro spouštění typu FIRST nebo DEPTH nemá žádný program otevřenou frontu aplikací pro odebírání zpráv (tj. atribut lokální fronty **OpenInputCount** je nulový).

   Poznámka:

   a. Pro sdílené fronty platí speciální podmínky v případě, že pro více správců front jsou spuštěny monitory spouštěčů pro frontu. V této situaci platí, že pokud má jeden nebo více správců front otevřenou frontu pro sdílený vstup, jsou kritéria spouštěče v ostatních správcích front považována za *TriggerType* MQTT_FIRST a *TriggerMsgPriority* nula. Když všichni správci front zavřou frontu pro vstup, podmínky spouštěče se vrátí k podmínkám uvedeným v definici fronty.

      Příkladem scénáře ovlivněného touto podmínkou je více správců front QM1, QM2a QM3 s monitorem spouštěčů spuštěným pro frontu aplikací A. Zpráva dorazí na A splňující podmínky pro spuštění a zpráva spouštěče je generována v inicializační frontě. Monitor spouštěčů na systému QM1 získá zprávu spouštěče a spustí aplikaci. Spuštěná aplikace otevře frontu aplikací pro sdílený vstup. Od tohoto okamžiku jsou podmínky spouštěče pro frontu aplikací A vyhodnoceny jako *TriggerType* MQTT_FIRST a *TriggerMsgPriority* nula ve správcích front QM2 a QM3, dokud produkt QM1 nezavře frontu aplikací.

   b. Pro sdílené fronty je tato podmínka použita pro každého správce front. To znamená, že má-li být zpráva spouštěče generována pro frontu tímto správcem front, musí být hodnota *OpenInputCount* správce front pro frontu nulová. Má-li však některý správce front ve skupině sdílení front otevřenou frontu pomocí volby MQOO_INPUT_EXCLUSIVE, nebude pro tuto frontu generována žádná zpráva spouštěče žádným ze správců front ve skupině sdílení front.

      Ke změně způsobu vyhodnocování podmínek spouštěče dojde, když spuštěná aplikace otevře frontu pro vstup. Ve scénářích, kde je spuštěn pouze jeden monitor spouštěčů, mohou mít jiné aplikace stejný efekt, protože podobně otevírají frontu aplikací pro vstup. Nezáleží na tom, zda byla fronta aplikací otevřena aplikací, která je spuštěna monitorem spouštěčů, nebo jinou aplikací; je to skutečnost, že fronta je otevřena pro vstup v jiném správci front, který způsobí změnu kritérií spouštěče.

5. V systému IBM MQ for z/OSplatí, že pokud se jedná o frontu aplikací s atributem **Usage** nastaveným na hodnotu MQUS_NORMAL, požadavky na získání pro tuto frontu nejsou blorovány (tj. atribut fronty **InhibitGet** je MQQA_GET_ALLOWED). Je-li spuštěna fronta aplikací s atributem **Usage** nastaveným na hodnotu MQUS_XMITQ, nejsou požadavky na získání pro tuto frontu blokovány.
6. Proveďte jednu z následujících akcí:
   - Atribut lokální fronty **ProcessName** pro frontu není prázdný a objekt definice procesu identifikovaný tímto atributem byl vytvořen, nebo
   - Atribut lokální fronty **ProcessName** pro frontu je prázdný, ale fronta je přenosová fronta. Vzhledem k tomu, že definice procesu je volitelná, může atribut **TriggerData** také obsahovat název kanálu, který má být spuštěn. V tomto případě zpráva spouštěče obsahuje 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 uvedena v atributu lokální fronty **InitiationQName** . Také:
   - Požadavky na získání nejsou pro inicializační frontu blokovány (tj. hodnota atributu fronty **InhibitGet** je MQQA_GET_ALLOWED).
   - Pro inicializační frontu nesmí být blokováno vkládání požadavků (tj. hodnota atributu fronty **InhibitPut** musí být MQQA_PUT_ALLOWED).
   - Hodnota atributu **Usage** inicializační fronty musí být MQUS_NORMAL.

- V prostředích, kde jsou podporovány dynamické fronty, nesmí být inicializační fronta dynamickou frontou, která byla označena jako logicky odstraněná.

8. Monitor spouštěčů má momentálně otevřenou inicializační frontu pro odebrání zpráv (tj. atribut lokální fronty **OpenInputCount** je větší než nula).
9. Řízení spouštěče (atribut lokální fronty**TriggerControl**) pro frontu aplikací je nastaveno na hodnotu MQTC_ON. Chcete-li to provést, nastavte atribut **trigger** při definování fronty nebo použijte příkaz ALTER QLOCAL.
10. Typ spouštěče (atribut lokální fronty**TriggerType**) není MQTT_NONE.

    Jsou-li splněny všechny požadované podmínky a zpráva, která způsobila podmínku spouštěče, je vložena jako součást pracovní jednotky, nebude zpráva spouštěče k dispozici pro načtení aplikací monitoru spouštěčů, dokud nebude transakce dokončena, bez ohledu na to, zda je transakce potvrzena, nebo v případě spouštěče typu MQTT_FIRST nebo MQTT_DEPTH, vrácena zpět.
11. Do fronty je umístěna vhodná zpráva pro **TriggerType** MQTT_FIRST nebo MQTT_DEPTH a fronta:

    - Nebyla dříve prázdná (MQTT_FIRST), nebo
    - Měl **TriggerDepth** nebo více zpráv (MQTT_DEPTH)

    a podmínky “2” na stránce 842 až “10” na stránce 844 (kromě “3” na stránce 842) jsou splněny, pokud v případě MQTT_FIRST uplynul od zápisu poslední zprávy spouštěče pro tuto frontu dostatečný interval (atribut správce front**TriggerInterval**).

    To je povoleno pro server front, který bude ukončen před zpracováním všech zpráv ve frontě. Účelem intervalu spouštěče je snížit počet generovaných duplicitních zpráv spouštěče.

    Poznámka: Pokud zastavíte a restartujete správce front, bude časovač **TriggerInterval** resetován. K dispozici je malé okno, během kterého je možné vytvořit dvě spouštěcí zprávy. Okno existuje, když je atribut spouštěče fronty nastaven na hodnotu Povoleno současně s doručením zprávy a fronta nebyla dříve prázdná (MQTT_FIRST) nebo měla **TriggerDepth** nebo více zpráv (MQTT_DEPTH).
12. Jediná aplikace obsluhující frontu zadá volání MQCLOSE pro **TriggerType** MQTT_FIRST nebo MQTT_DEPTH a existuje alespoň:

    - jedna (MQTT_FIRST) nebo
    - **TriggerDepth** (MQTT_DEPTH)

    zprávy ve frontě s dostatečnou prioritou (podmínka “2” na stránce 842) a podmínky “6” na stránce 843 až “10” na stránce 844 jsou také splněny.

    Tato volba umožňuje serveru front, který zadá volání MQGET, najít frontu prázdnou, a tak ji ukončí. V intervalu mezi voláním MQGET a MQCLOSE však dorazí jedna nebo více zpráv.

    Poznámka:

    a. Pokud program obsluhující frontu aplikací nenačte všechny zprávy, může to způsobit uzavřenou smyčku. Pokaždé, když program frontu zavře, správce front vytvoří další zprávu spouštěče, která způsobí, že monitor spouštěčů znovu spustí program serveru.

    b. Pokud program obsluhující frontu aplikací před uzavřením fronty odvolá svůj požadavek na získání (nebo pokud program ukončí nečinnost), stane se totéž. Pokud však program před zálohováním požadavku na získání uzavře frontu a fronta je jinak prázdná, nevytvoří se žádná zpráva spouštěče.

    c. Chcete-li zabránit výskytu takového cyklu, použijte pole *BackoutCount* deskriptoru MQMD ke zjištění zpráv, které jsou opakovaně vráceny zpět. Další informace viz téma “Zprávy, které jsou vráceny zpět” na stránce 44.
13. Následující podmínky jsou splněny pomocí příkazu MQSET nebo příkazu:

    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 existuje alespoň:

- jedna (MQTT_FIRST nebo MQTT_EVERY), nebo
- **TriggerDepth** (MQTT_DEPTH)

zprávy ve frontě s dostatečnou prioritou (podmínka “2” na stránce 842) a podmínky “4” na stránce 843 až “10” na stránce 844 (kromě “8” na stránce 844) jsou také spokojeni.

To umožní aplikaci nebo operátorovi změnit spouštěcí kritéria, když jsou podmínky pro spuštění spouštěče již splněny.

b. Hodnota atributu fronty **InhibitPut** inicializační fronty se změní z hodnoty MQQA_PUT_INHIBITED na hodnotu MQQA_PUT_ALLOWED a existuje alespoň:

- jedna (MQTT_FIRST nebo MQTT_EVERY), nebo
- **TriggerDepth** (MQTT_DEPTH)

zprávy s dostatečnou prioritou (podmínka “2” na stránce 842) ve všech frontách, pro které se jedná o inicializační frontu, a podmínky “4” na stránce 843 až “10” na stránce 844 jsou také splněny. (Pro každou takovou frontu je generována jedna zpráva spouštěče, která splňuje podmínky.)

To má umožnit, aby se negenerovaly zprávy spouštěče kvůli podmínce MQQA_PUT_INHIBITED v inicializační frontě, ale tato podmínka byla nyní změněna.

c. Hodnota atributu fronty **InhibitGet** fronty aplikací se změní z hodnoty MQQA_GET_INHIBITED na hodnotu MQQA_GET_ALLOWED a existuje alespoň:

- jedna (MQTT_FIRST nebo MQTT_EVERY), nebo
- **TriggerDepth** (MQTT_DEPTH)

zprávy s dostatečnou prioritou (podmínka “2” na stránce 842) ve frontě a podmínky “4” na stránce 843 až “10” na stránce 844, kromě “5” na stránce 843, jsou také splněny.

To umožňuje spouštět aplikace pouze v případě, že mohou načítat zprávy z fronty aplikací.

d. Aplikace monitoru spouštěčů vydá volání MQOPEN pro vstup z inicializační fronty a existuje alespoň:

- jedna (MQTT_FIRST nebo MQTT_EVERY), nebo
- **TriggerDepth** (MQTT_DEPTH)

zprávy s dostatečnou prioritou (podmínka “2” na stránce 842) ve všech aplikačních frontách, pro které se jedná o inicializační frontu, a podmínky “4” na stránce 843 až “10” na stránce 844 (kromě “8” na stránce 844) jsou také splněny, a žádná jiná aplikace nemá inicializační frontu otevřenou pro vstup (jedna spouštěcí zpráva je generována pro každou takovou frontu splňující podmínky).

To umožňuje zprávy přicházející do front v době, kdy není spuštěn monitor spouštěčů, a restartování správce front a ztrátu zpráv spouštěče (které jsou přechodné).

14. MSGDLVSQ je správně nastaven. Nastavíte-li MSGDLVSQ=FIFO, zprávy se doručí do fronty na základě Prvního prvního výstupu. Priorita zprávy je ignorována a předvolená priorita fronty je přiřazena ke zprávě. Je-li 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, dojde ke spuštění pro typ FIRST, EVERY a DEPTH. Chcete-li získat informace o těchto typech, prohlédněte si popis pole **TriggerType** pod “Řízení událostí spouštěče” na stránce 846.

Nastavíte-li hodnotu MSGDLVSQ=PRIORITY a priorita zprávy je větší nebo rovna hodnotě v poli *TriggerMsgPriority* , budou se zprávy započítávat pouze do události spouštěče. V tomto případě dojde ke spuštění pro typ FIRST, EVERY a DEPTH. Pokud například vložíte 100 zpráv s nižší prioritou než **TriggerMsgPriority**, efektivní hloubka fronty pro účely spuštění bude stále nulová. Pokud pak do fronty vložíte další zprávu, ale tentokrát je priorita větší nebo rovna **TriggerMsgPriority**, efektivní hloubka fronty se zvýší z nuly na jednu a podmínka pro **TriggerType** FIRST je splněna.

Notes:

1. Z kroku “12” na stránce 844 (kde jsou zprávy spouštěče generovány jako výsledek jiné události než zprávy přicházející do fronty aplikace) není zpráva spouštěče vložena jako součást pracovní jednotky. Je-li **TriggerType** také MQTT_EVERY a ve frontě aplikací je jedna nebo více zpráv, vygeneruje se pouze jedna zpráva spouštěče.
2. Pokud IBM MQ během operace MQPUT segmentuje zprávu, událost spouštěče nebude zpracována, dokud nebudou všechny segmenty úspěšně umístěny do fronty. Jakmile jsou však segmenty zpráv ve frontě, produkt IBM MQ je považuje za jednotlivé zprávy pro účely spouštění. Například jedna logická zpráva rozdělená na tři části způsobí, že při prvním MQPUT a segmentaci bude zpracována pouze jedna spouštěcí událost. Avšak každý ze tří segmentů způsobí, že se jejich vlastní spouštěcí události zpracují, jakmile se přesunou přes síť IBM MQ .
3. Je-li v systému IBM MQ for z/OSnastavena sdílená fronta pro spouštění a připojení k prostředku Coupling Facility, který je hostitelem sdílené fronty, je ztracena, může být vygenerována událost spouštěče a do inicializační fronty vložena zpráva. K tomu může dojít i v případě, že do původního nastavení sdílené fronty nebyla vložena žádná zpráva pro spuštění. To je způsobeno nadměrnou indikací bitů makrem IXLVECTR, jak je uvedeno v části Vektor oznámení seznamu.

Řízení událostí spouštěče

Události spouštěče řídíte pomocí některých atributů, které definují vaši frontu aplikací. Tyto informace také uvádějí příklady použití typů spouštěčů: KAŽDÝ, PRVNÍ a HLOUBKA.

Můžete povolit nebo zakázat spouštění a můžete vybrat počet nebo prioritu zpráv, které se započítávají do události spouštěče. Úplný popis těchto atributů je uveden v části Atributy objektů.

Příslušné atributy jsou:

TriggerControl
Pomocí tohoto atributu můžete povolit nebo zakázat spouštění pro frontu aplikací.

TriggerMsgPriority
Minimální priorita, kterou musí mít zpráva, aby se mohla počítat k události spouštěče. Pokud do fronty aplikací dorazí zpráva s prioritou nižší než *TriggerMsgPriority* , správce front tuto zprávu ignoruje při určování, zda má být vytvořena zpráva spouštěče. Je-li parametr *TriggerMsgPriority* nastaven na nulu, všechny zprávy se započítávají do události spouštěče.

TriggerType
Kromě typu spouštěče NONE (který zakazuje spouštění stejně jako nastavení parametru *TriggerControl* na hodnotu OFF) můžete použít následující typy spouštěčů k nastavení citlivosti fronty pro spouštění událostí:

Každý
Událost spouštěče se vyskytne pokaždé, když zpráva dorazí do fronty aplikace. Tento typ spouštěče použijte, chcete-li spustit více instancí aplikace.

FIRST
Událost spouštěče se vyskytne pouze v případě, že se počet zpráv ve frontě aplikací změní z nuly na jednu. Tento typ spouštěče použijte, pokud chcete, aby se obslužný program spustil, když první zpráva dorazí do fronty, pokračujte, dokud nebudou žádné další zprávy ke zpracování, a pak ukončete. Frontu musíte vždy zpracovat, dokud nebude prázdná. Další informace najdete v tématu “Speciální případ spouštěče typu FIRST” na stránce 847.

DEPTH
Událost spouštěče se vyskytne pouze v případě, že počet zpráv ve frontě aplikací dosáhne hodnoty atributu **TriggerDepth** . Typickým použitím tohoto typu spouštěče je spuštění programu, když jsou přijaty všechny odpovědi na sadu požadavků.

Spouštění podle hloubky: Při spouštění podle hloubky správce front zakáže spouštění (pomocí atributu *TriggerControl*) poté, co vytvoří zprávu spouštěče. Aplikace musí znovu povolit spouštění sama sebe (pomocí volání MQSET) poté, co k tomu došlo.

Akce zakázání spouštěče není pod řízením synchronizačního bodu, takže spuštění nelze znovu povolit zálohováním pracovní jednotky. Pokud program zálohuje požadavek vložení, který způsobil událost spouštěče, nebo pokud je program neukončen, musíte znovu povolit spuštění pomocí volání MQSET nebo příkazu ALTER QLOCAL.

TriggerDepth
Počet zpráv ve frontě, které způsobují událost spouštěče při použití spouštěče podle hloubky.

Podmínky, které musí být splněny, aby mohl správce front vytvořit zprávu spouštěče, jsou popsány v tématu “Podmínky pro událost spouštěče” na stránce 842.

Příklad použití spouštěče typu EVERY

Zvažte aplikaci, která generuje požadavky na pojištění motorových vozidel. Aplikace může odesílat zprávy požadavků řadě pojišťovacích společností, přičemž pokaždé zadá stejnou odpověď do fronty. V této frontě pro odpověď může být nastaven spouštěč typu EVERY, takže při každém přijetí odpovědi může odpověď spustit instanci serveru pro zpracování odpovědi.

Příklad použití typu spouštěče FIRST

Vezměme si organizaci s řadou poboček, které každý z nich předá podrobnosti o dnech podnikání do ústředí. Všichni to dělají ve stejnou dobu, na konci pracovního dne a v ústředí je aplikace, která zpracovává podrobnosti ze všech poboček. První zpráva, která dorazí do ústředí, může způsobit událost spouštěče, která spustí tuto aplikaci. Tato aplikace bude pokračovat ve zpracování, dokud ve frontě nebudou žádné další zprávy.

Příklad použití spouštěče typu DEPTH

Zvažte aplikaci cestovní kanceláře, která vytvoří jediný požadavek na potvrzení rezervace letu, potvrzení rezervace hotelového pokoje, pronájem auta a objednání některých cestovních šeků. Aplikace může tyto položky rozdělit do čtyř zpráv požadavků a odeslat je do samostatného místa určení. Může nastavit spouštěč typu DEPTH ve své frontě pro odpověď (s hloubkou nastavenou na hodnotu 4), aby byl restartován pouze po obdržení všech čtyř odpovědí.

Pokud do fronty pro odpověď dorazí jiná zpráva (pravděpodobně z jiného požadavku) před poslední ze čtyř odpovědí, aplikace, která žádost odeslala, se spustí brzy. Chcete-li se tomu vyhnout, při použití spouštěče DEPTH pro shromáždění více odpovědí na požadavek, vždy použijte novou frontu pro odpověď pro každý požadavek.

Speciální případ spouštěče typu FIRST

V případě spouštěče typu FIRST, pokud již existuje zpráva ve frontě aplikace při doručení jiné zprávy, správce front obvykle nevytvoří další zprávu spouštěče.

Aplikace obsluhující frontu však nemusí frontu ve skutečnosti otevřít (například aplikace může skončit, pravděpodobně kvůli problému se systémem). Pokud byl do objektu definice procesu vložen nesprávný název aplikace, aplikace obsluhující frontu nevybere žádnou ze zpráv. V těchto situacích, pokud do fronty aplikací dorazí jiná zpráva, není spuštěn žádný server, který by tuto zprávu zpracovával (a žádné další zprávy ve frontě).

Aby se s tím vypořádal, vytvoří správce front další zprávy spouštěče za následujících okolností:

- Pokud do fronty aplikací dorazí jiná zpráva, ale pouze v případě, že od doby, kdy správce front vytvořil poslední zprávu spouštěče pro tuto frontu, uplynul předdefinovaný časový interval. Tento časový interval je definován v atributu správce front *TriggerInterval*. Výchozí hodnota je 999 999 999 milisekund.
- V systému IBM MQ for z/OSjsou fronty aplikací, které pojmenují otevřenou inicializační frontu, pravidelně skenovány. Pokud od odeslání poslední zprávy spouštěče uplynulo *TRIGINT* milisekund a fronta splňuje podmínky pro událost spouštěče a hodnota CURDEPTH je větší než nula, vygeneruje se zpráva spouštěče. Tento proces se nazývá spouštění backstop.

Při rozhodování o hodnotě intervalu spouštěče, která má být použita ve vaší aplikaci, zvažte následující body:

- Pokud nastavíte parametr *TriggerInterval* na nízkou hodnotu a neexistuje žádná aplikace obsluhující frontu aplikací, typ spouštěče FIRST se může chovat jako typ spouštěče EVERY. To závisí

na rychlosti, jakou jsou zprávy vkládány do fronty aplikací, což může záviset na jiné aktivitě systému. Je tomu tak proto, že pokud je interval spouštěče velmi malý, je při každém vložení zprávy do fronty aplikací generována jiná zpráva spouštěče, i když typ spouštěče je FIRST, nikoli EVERY. (Typ spouštěče FIRST s intervalem spouštěče nula je ekvivalentní typu spouštěče EVERY.)

- Pokud v systému IBM MQ for z/OS nastavíte hodnotu *TRIGINT* na nízkou a neexistuje žádná aplikace obsluhující frontu aplikace typu spouštěče FIRST, spuštění backstop vygeneruje zprávu spouštěče pokaždé, když dojde k pravidelnému skenování front aplikací, které pojmenují otevřené inicializační fronty.
- Je-li jednotka práce vrácena zpět (viz Spouštěcí zprávy a jednotky práce). a interval spouštěče byl nastaven na vysokou hodnotu (nebo výchozí hodnotu), jedna zpráva spouštěče se vygeneruje, když se transakce odvolá. Pokud jste však nastavili interval spouštěče na nízkou hodnotu nebo na nulu (což způsobí, že se typ spouštěče FIRST bude chovat jako typ spouštěče EVERY), může být vygenerováno mnoho zpráv spouštěče. Pokud je transakce odvolána, všechny zprávy spouštěče jsou stále k dispozici. Počet generovaných zpráv spouštěče závisí na intervalu spouštěče. Je-li interval spouštěče nastaven na nulu, vygeneruje se maximální počet zpráv.

Návrh aplikace, která používá spuštěné fronty

Viděli jste, jak nastavit a řídit spouštění pro vaše aplikace. Zde je několik tipů, které je třeba zvážit při návrhu aplikace.

Spustit zprávy a jednotky práce

Zprávy spouštěče vytvořené kvůli událostem spouštěče, které nejsou součástí pracovní jednotky, jsou vloženy do inicializační fronty, mimo jakoukoli pracovní jednotku, bez závislosti na jiných zprávách, a jsou okamžitě k dispozici pro načtení monitorem spouštěče.

Zprávy spouštěče vytvořené kvůli událostem spouštěče, které jsou součástí pracovní jednotky, jsou zpřístupněny v inicializační frontě při vyřešení pracovní jednotky, bez ohledu na to, zda je transakce potvrzena nebo vrácena zpět.

Pokud se správci front nepodaří vložit zprávu spouštěče do inicializační fronty, bude vložena do fronty nedoručených zpráv (nedoručených zpráv).

Poznámka:

1. Správce front spočítá potvrzenou i nepotvrzenou zprávu při posuzování, zda existují podmínky pro událost spouštěče.

   Při spouštění typu FIRST nebo DEPTH jsou zprávy spouštěče k dispozici i v případě, že je jednotka práce vrácena zpět, takže zpráva spouštěče je vždy k dispozici, když jsou splněny požadované podmínky. Zvažte například požadavek na vložení v rámci pracovní jednotky pro frontu, která je spuštěna s typem spouštěče FIRST. To způsobí, že správce front vytvoří zprávu spouštěče. Pokud se vyskytne jiný požadavek vložení, z jiné pracovní jednotky to nezpůsobí další událost spouštěče, protože počet zpráv ve frontě aplikace se nyní změnil z jedné na dvě, což nesplňuje podmínky pro událost spouštěče. Nyní, pokud je první jednotka práce vrácena zpět, ale druhá je potvrzena, je stále vytvořena zpráva spouštěče.

   To však znamená, že zprávy spouštěče jsou někdy vytvářeny, když nejsou splněny podmínky pro událost spouštěče. Aplikace, které používají spouštění, musí být vždy připraveny na zpracování této situace. Doporučuje se použít volbu wait s voláním MQGET a nastavit parametr *WaitInterval* na vhodnou hodnotu.

   Vytvořené zprávy spouštěče jsou vždy k dispozici bez ohledu na to, zda je jednotka práce vrácena zpět nebo potvrzena.
2. Pro lokální sdílené fronty (tj. sdílené fronty ve skupině sdílení front) počítá správce front pouze potvrzené zprávy.

Získávání zpráv ze spuštěné fronty

Při návrhu aplikací, které používají spouštění, mějte na paměti, že může dojít k prodlevě mezi spuštěním programu monitorem spouštěčů a zpřístupněním dalších zpráv ve frontě aplikací. K tomu může dojít, když je zpráva, která způsobuje událost spouštěče, potvrzena před ostatními.

Chcete-li povolit čas pro příchod zpráv, vždy použijte volbu wait, když použijete volání MQGET k odebrání zpráv z fronty, pro kterou jsou nastaveny podmínky spouštěče. Hodnota *WaitInterval* musí být dostatečná, aby umožňovala nejdelší rozumný čas mezi vloženou zprávou a potvrzeným voláním vložení. Pokud zpráva přichází ze vzdáleného správce front, bude tento čas ovlivněn:

- Počet zpráv vložených před potvrzením
- Rychlost a dostupnost komunikačního spojení
- Velikost zpráv

Pro příklad situace, kdy byste měli použít volání MQGET s volbou wait, zvažte stejný příklad, který jsme použili při popisu pracovních jednotek. Jedná se o požadavek vložení v rámci pracovní jednotky pro frontu, která je spuštěna 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 další událost spouštěče, protože počet zpráv ve frontě aplikací se nezměnil z nuly na jednu. Nyní, pokud je první jednotka práce vrácena zpět, ale druhá je potvrzena, je stále vytvořena zpráva spouštěče. Zpráva spouštěče se tedy vytvoří v době, kdy je první jednotka práce vrácena zpět. Pokud dojde k významné prodlevě před potvrzením druhé zprávy, spuštěná aplikace na ni bude možná muset počkat.

Při spuštění typu DEPTH může dojít k prodlevě, i když jsou všechny příslušné zprávy nakonec potvrzeny. Předpokládejme, že atribut fronty **TriggerDepth** má hodnotu 2. Když do fronty dorazí dvě zprávy, druhá způsobí, že se vytvoří zpráva spouštěče. Je-li však druhá zpráva první, která má být potvrzena, je to v té době, kdy je zpráva spouštěče k dispozici. Monitor spouštěčů spustí program serveru, ale program může načíst pouze druhou zprávu, dokud není potvrzena první zpráva. Program tedy možná bude muset čekat na zpřístupnění první zprávy.

Navrhněte aplikaci tak, aby byla ukončena, pokud nejsou k dispozici žádné zprávy pro načtení po vypršení intervalu čekání. Pokud jedna nebo více zpráv dorazí později, spoléhejte na to, že vaše aplikace bude při jejich zpracování znovu spuštěna. Tato metoda zabraňuje nečinnosti aplikací a zbytečnému používání prostředků.

Zpracování inicializační fronty monitory spouštěčů

Pro správce front je monitor spouštěčů podobný jakékoli jiné aplikaci, která obsluhuje frontu. Monitor spouštěčů však obsluhuje inicializační fronty.

Monitor spouštěčů je obvykle nepřetržitě běžící program. Když zpráva spouštěče dorazí do inicializační fronty, monitor spouštěčů tuto zprávu načte. Používá informace ve zprávě k zadání příkazu ke spuštění aplikace, která má zpracovat zprávy ve frontě aplikace.

Monitor spouštěčů musí předat programu, který spouští, dostatečné informace, aby mohl provádět správné akce ve správné aplikační frontě.

Iniciátor kanálu je příkladem speciálního typu monitoru spouštěčů pro agenty kanálů zpráv. V této situaci však musíte použít buď typ spouštěče FIRST, nebo DEPTH.

ALW *Monitory spouštěčů na systémech AIX, Linux, and Windows*

Toto téma obsahuje informace o monitorech spouštěčů poskytovaných na systémech AIX, Linux, and Windows .

Pro prostředí serveru jsou k dispozici následující monitory spouštěčů:

amqstrg0
Jedná se o ukázkový monitor spouštěčů, který poskytuje podmnožinu funkce poskytované příkazem **runmqtrm**. Další informace o amqstrg0viz “Použití ukázkových programů na platformě Multiplatforms” na stránce 1024 .

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. Volá programy pro příslušné zprávy spouštěče. Tento monitor spouštěčů podporuje výchozí typ aplikace.

Příkazový řetězec předaný monitorem spouštěčů operačnímu systému je sestaven takto:

1. *ApplId* z příslušné definice PROCESS (je-li vytvořena)
2. Struktura MQTMC2 uzavřená v uvozovkách.
3. *EnvData* z příslušné definice PROCESS (je-li vytvořena)

kde *ApplId* je název programu, který se má spustit tak, jak by měl být zadán na příkazovém řádku.

Předaným parametrem je znaková struktura MQTMC2 . Je vyvolán příkazový řetězec, který má tento řetězec, přesně tak, jak je uveden, v uvozovkách, aby jej systémový příkaz přijal jako jeden parametr.

Monitor spouštěčů nehledá, zda je v inicializační frontě další zpráva, dokud není dokončena aplikace, kterou právě spustila. Pokud má aplikace mnoho zpracování, nemusí být monitor spouštěčů schopen držet krok s počtem příchozích zpráv spouštěče. Máte dvě možnosti:

- Mít spuštěno více monitorů spouštěčů
- Spustit spuštěné aplikace na pozadí

Máte-li spuštěno více monitorů spouštěčů, můžete řídit maximální počet aplikací, které lze spustit najednou. Pokud spouštíte aplikace na pozadí, neexistuje žádné omezení stanovené produktem IBM MQ na počet aplikací, které lze spustit.

Linux AIX Chcete-li spustit spuštěnou aplikaci na pozadí v systému AIX and Linux, vložte & na konec *EnvData* definice PROCESS.

Chcete-li spustit spuštěnou aplikaci na pozadí na systémech Windows , v poli *ApplId* zadejte před název aplikace příkaz START. Příklad:

```
START ?B AMQSECHA
```

Poznámka: Windows Pokud má cesta Windows mezery jako součást názvu cesty, měly by být uzavřeny v uvozovkách (") aby se zajistilo, že se s ním zachází jako s jediným argumentem. Například "`C:\Program Files\Application Directory\Application.exe`".

Následuje příklad řetězce APPLICID, kde název souboru obsahuje jako součást cesty mezery:

```
START "" /B "C:\Program Files\Application Directory\Application.exe"
```

Syntaxe příkazu Windows START v příkladu zahrnuje prázdný řetězec uzavřený v uvozovkách. Hodnota START určuje, že první argument v uvozovkách bude považován za název nového příkazu. Chcete-li se ujistit, že produkt Windows nezaměňuje cestu k aplikaci pro argument 'title', přidejte do příkazu před název aplikace řetězec názvu uzavřený v uvozovkách.

Pro klienta IBM MQ jsou k dispozici následující monitory spouštěčů:

runmqtmc

Jedná se o totéž jako příkaz runmqtrm s tím rozdílem, že je propojen s knihovnami IBM MQ MQI client .

ALW *Monitor spouštěčů pro CICS*

Monitor spouštěčů amqltmc0 je poskytován pro CICS. Funguje stejně jako standardní monitor spouštěčů runmqtrm, ale spustíte jej jiným způsobem a spustí transakce CICS .

Toto téma se týká pouze systémů Windows, AIX and Linux x86-64 .

Monitor spouštěčů je dodáván jako program CICS ; definujte jej s názvem transakce o délce 4 znaků. Zadejte 4znakový název pro spuštění monitoru spouštěčů. Používá výchozího správce front (jak je uvedeno v souboru qm.ini nebo v systému IBM MQ for Windowsv registru) a SYSTEM.CICS.INITIATION.QUEUE.

Chcete-li použít jiného správce front nebo frontu, sestavte strukturu monitoru spouštěčů MQTMC2 . To vyžaduje, abyste napsali program pomocí volání EXEC CICS START, protože struktura je příliš dlouhá na to, aby ji bylo možné přidat jako parametr. Poté předejte strukturu MQTMC2 jako data požadavku START pro monitor spouštěčů.

Používáte-li strukturu MQTMC2 , musíte do monitoru spouštěčů zadat pouze parametry *StrucId*, *Version*, *QName*a **QMgrName** , protože neodkazuje na žádná jiná pole.

Zprávy se načítají z inicializační fronty a používají se ke spuštění transakcí CICS pomocí příkazu EXEC CICS START, za předpokladu, že parametr APPL_TYPE ve zprávě spouštěče je MQAT_CICS. Čtení zpráv z inicializační fronty se provádí pod řízením synchronizačního bodu CICS .

Zprávy se generují při spuštění a zastavení monitoru a při výskytu chyby. Tyto zprávy jsou odeslány do dočasné datové fronty CSMT.

Tabulka 129. Dostupné verze monitoru spouštěčů.

Tabulka se dvěma sloupci. první sloupec uvádí dostupné verze monitoru spouštěčů a druhý sloupec uvádí, pro které platformy je každá verze použita.

Verze	**Použijte**
amqltmc0	TXSeries pro: • AIX • Linux x86-64 systémy
amqltmc4	Windows TXSeries pro Windows 5.1
amqltmcc	Klientská vázaná verze monitoru spouštěčů CICS

Potřebujete-li monitor spouštěčů pro jiná prostředí, napište program, který dokáže zpracovat zprávy spouštěčů, které správce front vkládá do inicializačních front. Takový program by měl provádět následující akce:

1. Volání MQGET slouží k čekání na doručení zprávy do inicializační fronty.
2. Zkontrolujte pole struktury MQTM zprávy spouštěče a vyhledejte název aplikace, která se má spustit, a prostředí, ve kterém je spuštěna.
3. Zadejte spouštěcí příkaz specifický pro dané prostředí.

   z/OS Například v dávce z/OS odešlete úlohu do interního čtecího programu.
4. V případě potřeby převeďte strukturu MQTM na strukturu MQTMC2 .
5. Předejte spuštěnou aplikaci buď strukturu MQTMC2 , nebo strukturu MQTM. Může obsahovat uživatelská data.
6. Přidružte k frontě aplikace aplikaci, která má tuto frontu obsluhovat. To provedete pojmenováním objektu definice procesu (je-li vytvořen) v atributu **ProcessName** fronty. Chcete-li pojmenovat objekt definice procesu, můžete použít příkaz **DEFINE QLOCAL** nebo **ALTER QLOCAL** .

   IBM i V systému IBM imůžete také použít CRTMQMQ nebo CHGMQMQ.

Další informace o rozhraní monitoru spouštěčů viz MQTMC2.

IBM i *Monitory spouštěčů na systému IBM i*

V systému IBM ipoužijte místo řídicího příkazu **runmqtrm** příkaz IBM MQ for IBM i CL **STRMQMTRM**.

Příkaz STRMQMTRM použijte takto:

```
STRMQMTRM INITQNAME(InitQ) MQMNAME(QMgrName)
```

Podrobnosti jsou jako pro runmqtrm.

K dispozici jsou také následující ukázkové programy, které můžete použít jako modely pro psaní vlastních monitorů spouštěčů:

AMQSTRG4
Jedná se o monitor spouštěčů, který zadá úlohu IBM i pro proces, který má být spuštěn, ale to znamená, že ke každé zprávě spouštěče je přidruženo další zpracování.

AMQSERV4
Jedná se o spouštěcí server. Pro každou zprávu spouštěče tento server spustí příkaz pro proces ve své vlastní úloze a může volat transakce CICS .

Monitor spouštěčů i server spouštěčů předávají spouštěným programům strukturu MQTMC2 . Popis této struktury viz MQTMC2. Obě tyto ukázky jsou dodávány ve zdrojovém i spustitelném tvaru.

Protože tyto monitory spouštěčů mohou vyvolat pouze nativní programy IBM i , nemohou spouštět programy Java přímo, protože třídy Java jsou umístěny v IFS. Avšak programy Java mohou být spouštěny nepřímo spuštěním programu CL, který pak vyvolá program Java a projde přes strukturu TMC2 . Minimální velikost struktury TMC2 je 732 bajtů.

Zde je zdroj ukázkového příkazového procesoru:

```
PGM PARM(&TMC2)
  DCL &TMC2 *CHAR LEN(800)
    ADDENVVAR ENVVAR(TM) VALUE(&TMC2)
    QSH  CMD('java_pgmname $TM')
    RMVENVVAR ENVVAR(TM)
ENDPGM
```

Pro produkt IBM MQ MQI clientje k dispozici následující program monitoru spouštěčů: RUNMQTMC

Vyvolejte příkaz RUNMQTMC následujícím způsobem:

```
CALL PGM(QMQM/RUNMQTMC) PARM('-m' QMgrName '-q' InitQ)
```

Vlastnosti zpráv spouštěče

Následující témata popisují některé další vlastnosti zpráv spouštěče.

- “Perzistence a priorita zpráv spouštěče” na stránce 852
- “Zprávy o restartování a spouštění správce front” na stránce 853
- “Spustit zprávy a změny atributů objektu” na stránce 853
- “Formát zpráv spouštěče” na stránce 853

Perzistence a priorita zpráv spouštěče

Zprávy spouštěče nejsou trvalé, protože pro ně neexistuje žádný požadavek.

Podmínky pro generování spouštěcích událostí však přetrvávají, takže se zprávy spouštěče generují vždy, když jsou tyto podmínky splněny. Dojde-li ke ztrátě zprávy spouštěče, další existence zprávy aplikace ve frontě aplikací zaručuje, že správce front vygeneruje zprávu spouštěče, jakmile jsou splněny všechny podmínky.

Pokud je transakce odvolána, všechny zprávy spouštěče, které vygenerovala, jsou vždy doručeny.

Zprávy spouštěče mají výchozí prioritu inicializační fronty.

Zprávy o restartování a spouštění správce front

Po restartování správce front může být při dalším otevření inicializační fronty pro vstup do této inicializační fronty vložena zpráva spouštěče, pokud má přidružená fronta aplikací zprávy a je definována pro spouštění.

Spustit zprávy a změny atributů objektu

Zprávy spouštěče jsou vytvářeny podle hodnot atributů spouštěče platných v době události spouštěče.

Pokud není zpráva spouštěče zpřístupněna monitoru spouštěče až později (protože zpráva, která způsobila její vygenerování, byla vložena do pracovní jednotky), žádné změny atributů spouštěče v mezidobí nemají žádný vliv na zprávu spouštěče. Zakázání spouštěče zejména nebrání tomu, aby byla zpráva spouštěče zpřístupněna po jejím vytvoření. Také fronta aplikací již nemusí existovat v době, kdy je zpráva spouštěče zpřístupněna.

Formát zpráv spouštěče

Formát zprávy spouštěče je definován strukturou MQTM.

Toto pole obsahuje následující pole, která správce front vyplní při vytváření zprávy spouštěče s použitím informací v definicích objektů fronty aplikací a procesu přidruženého k této frontě:

StrucId
Identifikátor struktury.

Version
Verze struktury.

QName
Název fronty aplikací, ve které došlo k události spouštěče. Když správce front vytvoří zprávu spouštěče, vyplní toto pole pomocí atributu **QName** fronty aplikací.

ProcessName
Název objektu definice procesu, který je přidružen k frontě aplikace. Když správce front vytvoří zprávu spouštěče, vyplní toto pole pomocí atributu **ProcessName** fronty aplikací.

TriggerData
Pole volného formátu pro použití monitorem spouštěčů. Když správce front vytvoří zprávu spouštěče, vyplní toto pole pomocí atributu **TriggerData** fronty aplikací. V libovolném IBM MQ produktu kromě IBM MQ for z/OSlze toto pole použít k určení 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** objektu definice procesu identifikovaného v souboru *ProcessName*.

ApplId
Znakový řetězec, který identifikuje aplikaci, kterou má monitor spouštěčů spustit. Když správce front vytvoří zprávu spouštěče, vyplní toto pole pomocí atributu **ApplId** objektu definice procesu identifikovaného v souboru *ProcessName*.

Když použijete monitor spouštěčů CKTI dodaný produktem CICS, atribut **ApplId** objektu definice procesu je identifikátor transakce CICS .

Když použijete CSQQTRMN dodanou produktem IBM MQ for z/OS, atribut **ApplId** objektu definice procesu je identifikátor transakce IMS .

EnvData
Znakové pole obsahující data související s prostředím pro použití monitorem spouštěčů. Když správce front vytvoří zprávu spouštěče, vyplní toto pole pomocí atributu **EnvData** objektu definice procesu identifikovaného v souboru *ProcessName*. Monitor spouštěčů dodávaný s produktem CICS(CKTI) nebo monitor spouštěčů dodávaný s produktem IBM MQ for z/OS(CSQQTRMN) toto pole nepoužívají, ale jiné monitory spouštěčů jej mohou použít.

UserData
Znakové pole obsahující uživatelská data pro použití monitorem spouštěčů. Když správce front vytvoří zprávu spouštěče, vyplní toto pole pomocí atributu **UserData** objektu definice procesu identifikovaného 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 spuštění nefunguje

Program se nespustí, pokud monitor spouštěčů nemůže spustit program nebo správce front nemůže doručit zprávu spouštěče. Například aplikace v objektu procesu musí uvádět, že program se má spustit na pozadí; jinak monitor spouštěčů nemůže spustit program.

Pokud je vytvořena zpráva spouštěče, ale nelze ji vložit do inicializační fronty (například protože fronta je plná nebo délka zprávy spouštěče je větší než maximální délka zprávy uvedená pro inicializační frontu), zpráva spouštěče je místo toho vložena do fronty nedoručených zpráv (nedoručených zpráv).

Pokud operaci vložení do fronty nedoručených zpráv nelze úspěšně dokončit, zpráva spouštěče se zahodí a zpráva s varováním se odešle z/OS na konzolu z/OS nebo systémovému operátorovi nebo se vloží do protokolu chyb.

Vložení zprávy spouštěče do fronty nedoručených zpráv může generovat zprávu spouštěče pro tuto frontu. Tato druhá zpráva spouštěče je vyřazena, pokud přidá zprávu do fronty nedoručených zpráv.

Pokud je program úspěšně spuštěn, ale před přijetím zprávy z fronty se neukončí, použijte obslužný program trasování (například CICS AUXTRACE, pokud je program spuštěn pod CICS). najít příčinu selhání.

Práce s rozhraním MQI a klastry

Existují speciální volby pro volání a návratové kódy, které se týkají klastrování.

Pomocí následujících odkazů získáte další informace o volbách, které jsou k dispozici pro volání a návratové kódy pro použití s klastry:

- “MQOPEN a klastry” na stránce 855
- “MQPUT, MQPUT1 a klastry” na stránce 856
- “MQINQ a klastry” na stránce 856
- “MQSET a klastry” na stránce 857
- “Návratové kódy” na stránce 857

Související pojmy

“Přehled rozhraní fronty zpráv” na stránce 700
Informace o komponentách rozhraní MQI (Message Queue Interface).

“Připojení ke správci front a odpojení od něj” na stránce 713
Chcete-li používat programovací služby IBM MQ , musí mít program připojení ke správci front. Pomocí těchto informací se dozvíte, jak se připojit ke správci front a jak se od něj odpojit.

“Otevírání a zavírání objektů” na stránce 720
Tyto informace poskytují náhled na otevírání a zavírání objektů IBM MQ .

“Vkládání zpráv do fronty” na stránce 730
Pomocí těchto informací se dozvíte, jak vkládat zprávy do fronty.

“Získávání zpráv z fronty” na stránce 744
Pomocí těchto informací získáte informace o získávání zpráv z fronty.

“Zjišťování a nastavení atributů objektu” na stránce 822
Atributy jsou vlastnosti, které definují charakteristiky objektu IBM MQ .

“Potvrzení a zálohování jednotek práce” na stránce 825
Tyto informace popisují, jak potvrdit a vrátit zpět všechny obnovitelné operace get a put, které se vyskytly v transakci.

“Spuštění aplikací IBM MQ pomocí spouštěčů” na stránce 836
Získejte informace o spouštěčích a o tom, jak spouštět aplikace IBM MQ pomocí spouštěčů.

“Používání a psaní aplikací na systému IBM MQ for z/OS” na stránce 858
Aplikace IBM MQ for z/OS lze vytvořit z programů, které jsou spuštěny v mnoha různých prostředích. To znamená, že mohou využívat zařízení, která jsou k dispozici ve více než jednom prostředí.

“IMS a IMS přemostění aplikací na IBM MQ for z/OS” na stránce 66
Tyto informace vám pomohou psát aplikace IMS pomocí IBM MQ.

MQOPEN a klastry

Fronta, do které je zpráva vložena nebo z níž je čtena při otevření fronty klastru, závisí na volání MQOPEN .

Výběr cílové fronty

Pokud v deskriptoru objektu MQODnezadáte název správce front, správce front vybere správce front, do kterého má být zpráva odeslána. Zadáte-li název správce front v deskriptoru objektu, budou zprávy vždy odesílány vybranému správci front.

Pokud správce front vybírá cílového správce front, výběr závisí na volbách vazby MQOO_BIND_* a na tom, zda existuje lokální fronta. Pokud existuje lokální instance fronty, je vždy otevřena přednostně pro vzdálenou instanci, pokud není atribut CLWLUSEQ nastaven na hodnotu ANY. Jinak výběr závisí na volbách vazby. Buď MQOO_BIND_ON_OPEN , nebo MQOO_BIND_ON_GROUP musí být zadáno při použití skupin zpráv s klastry, aby se zajistilo, že všechny zprávy ve skupině budou zpracovány ve stejném místě určení.

Pokud správce front vybírá cílového správce front, učiní tak metodou round-robin s použitím algoritmu správy pracovní zátěže; viz Vyvažování pracovní zátěže v klastrech.

Při použití algoritmu vyrovnávání pracovní zátěže závisí na způsobu otevření fronty klastru:

- MQOO_BIND_ON_OPEN -algoritmus se použije jednou v době, kdy je fronta otevřena aplikací.
- MQOO_BIND_NOT_FIXED -algoritmus se používá pro všechny zprávy vkládané do fronty.
- MQOO_BIND_ON_GROUP -algoritmus se používá jednou na začátku každé skupiny zpráv.

MQOO_BIND_ON_OPEN
Volba MQOO_BIND_ON_OPEN ve volání MQOPEN určuje, že má být opraven cílový správce front. Volbu MQOO_BIND_ON_OPEN použijte, pokud je v klastru více instancí stejné fronty. Všechny zprávy vkládané do fronty, které určují popisovač objektu vrácený voláním funkce MQOPEN , jsou směrovány do stejného správce front.

- Použijte volbu MQOO_BIND_ON_OPEN , pokud mají zprávy afinitu. Pokud má být například dávka zpráv zpracována stejným správcem front, zadejte při otevření fronty hodnotu MQOO_BIND_ON_OPEN . Produkt IBM MQ opraví správce front a trasu, kterou mají provést všechny zprávy 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 opraven. Zprávy zapsané do fronty určující popisovač objektu vrácený voláním funkce MQOPEN jsou směrovány do správce front v čase MQPUT na základě jednotlivých zpráv. Volbu MQOO_BIND_NOT_FIXED použijte v případě, že nechcete vynutit zápis všech zpráv do stejného místa určení.

- Nezadávejte současně MQOO_BIND_NOT_FIXED a MQMF_SEGMENTATION_ALLOWED . Pokud tak učiníte, segmenty zprávy mohou být doručeny různým správcům front, kteří jsou rozptýleni v rámci klastru.

MQOO_BIND_ON_GROUP
Umožňuje aplikaci požadovat, aby byla skupina zpráv přidělena ke stejné cílové instanci. Tato volba je platná pouze pro fronty a ovlivňuje pouze fronty klastru. Je-li zadána pro frontu, která není frontou klastru, volba se ignoruje.

- Skupiny jsou směrovány do jednoho místa určení pouze v případě, že je v příkazu MQPUT zadána hodnota MQPMO_LOGICAL_ORDER. Je-li zadána volba MQOO_BIND_ON_GROUP, ale zpráva není součástí logické skupiny, použije se místo toho chování BIND_NOT_FIXED.

MQOO_BIND_AS_Q_DEF
Pokud neuvedete MQOO_BIND_ON_OPEN, MQOO_BIND_NOT_FIXED nebo MQOO_BIND_ON_GROUP, výchozí volba je MQOO_BIND_AS_Q_DEF. Použití MQOO_BIND_AS_Q_DEF způsobí, že vazba, která se používá pro popisovač fronty, bude převzata z atributu fronty DefBind .

Relevance voleb MQOPEN

MQOPEN Volby MQOO_BROWSE, MQOO_INPUT_*nebo MQOO_SET vyžadují lokální instanci fronty klastru, aby byl produkt MQOPEN úspěšný.

MQOPEN Volby MQOO_OUTPUT, MQOO_BIND_*nebo MQOO_INQUIRE nevyžadují, aby byla lokální instance fronty klastru úspěšná.

Vyřešený název správce front

Je-li název správce front interpretován v čase MQOPEN , bude převedený název vrácen aplikaci. Pokud se aplikace pokusí použít tento název při následném volání MQOPEN , může zjistit, že nemá autorizaci pro přístup k názvu.

MQPUT, MQPUT1 a klastry

Je-li v systému MQOPEN zadána volba MQOO_BIND_NOT_FIXED , rutiny správy pracovní zátěže zvolí, které místo určení MQPUT nebo MQPUT1 vyberou.

Je-li pro volání MQOPEN zadána volba MQOO_BIND_NOT_FIXED , vyvolá každé následné volání funkce MQPUT rutinu správy pracovní zátěže s cílem určit, kterému správci front má být zpráva odeslána. Cíl a trasa, která má být provedena, jsou vybrány podle jednotlivých zpráv. Místo určení a přenosová cesta se mohou po vložení zprávy změnit, pokud se změní podmínky v síti. Volání MQPUT1 vždy funguje tak, jako by byla MQOO_BIND_NOT_FIXED v platnosti, to znamená, že vždy vyvolá rutinu správy pracovní zátěže.

Pokud rutina správy pracovní zátěže vybrala správce front, lokální správce front dokončí operaci vložení. Zprávu lze umístit do různých front:

1. Pokud je cílem lokální instance fronty, zpráva se umístí do lokální fronty.
2. Pokud je cílem správce front v klastru, je zpráva umístěna do přenosové fronty klastru.
3. Pokud je cílem správce front mimo klastr, je zpráva umístěna do přenosové fronty se stejným názvem jako cílový správce front.

Je-li ve volání MQOPEN zadán parametr MQOO_BIND_ON_OPEN , volání MQPUT nevyvolají rutinu správy pracovní zátěže, protože místo určení a trasa již byly vybrány.

MQINQ a klastry

Která fronta klastru je zjišťována, závisí na volbách, které kombinujete s produktem MQOO_INQUIRE.

Než se budete moci dotázat na frontu, otevřete ji pomocí volání MQOPEN a zadejte MQOO_INQUIRE.

Chcete-li se dotázat na frontu klastru, použijte volání MQOPEN a kombinujte další volby s volbou MQOO_INQUIRE. Atributy, které lze zjistit, závisí na tom, zda existuje lokální instance fronty klastru, a na tom, jak je fronta otevřena:

- Kombinace MQOO_BROWSE, MQOO_INPUT_*nebo MQOO_SET s MQOO_INQUIRE vyžaduje lokální instanci fronty klastru, aby bylo otevření úspěšné. V tomto případě můžete zjistit všechny atributy, které jsou platné pro lokální fronty.
- Kombinace MQOO_OUTPUT s MQOO_INQUIREa neuvedení žádné z předchozích voleb, otevřená instance je buď:
  - Instance v lokálním správci front, pokud existuje. V tomto případě můžete zjistit všechny atributy, které jsou platné pro lokální fronty.

– Instance jinde v klastru, pokud neexistuje žádná lokální instance správce front. V tomto případě lze zjišťovat pouze následující atributy. Atribut QType má v tomto případě hodnotu MQQT_CLUSTER .
  - DefBind
  - DefPersistence
  - DefPriority
  - InhibitPut
  - QDesc
  - QName
  - QTYPE

Chcete-li se dotázat na atribut DefBind fronty klastru, použijte volání MQINQ se selektorem 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 zadána hodnota MQBND_BIND_ON_OPEN nebo MQBND_BIND_ON_GROUP .

Chcete-li se dotázat na atributy CLUSTER a CLUSNL lokální instance fronty, použijte volání MQINQ se selektorem MQCA_CLUSTER_NAME nebo selektorem MQCA_CLUSTER_NAMELIST.

Poznámka: Pokud otevřete frontu klastru, aniž byste opravili frontu, ke které je produkt MQOPEN svázán, následná volání MQINQ se mohou dotázat na různé instance fronty klastru.

Související pojmy

“Volba MQOPEN pro frontu klastru” na stránce 725

Vazba použitá pro manipulátor fronty je převzata z atributu fronty **DefBind** , který může mít hodnotu MQBND_BIND_ON_OPEN, MQBND_BIND_NOT_FIXEDnebo MQBND_BIND_ON_GROUP.

MQSET a klastry

Volba MQOPEN option MQOO_SET vyžaduje lokální instanci fronty klastru, aby byl produkt MQSET úspěšný.

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 . Můžete nastavit atributy lokálního aliasu nebo vzdálené fronty. Nezáleží na tom, zda je cílová fronta frontou klastru definovanou v jiném správci front.

Návratové kódy

Návratové kódy specifické pro klastry

MQRC_CLUSTER_EXIT_ERROR (2266 X'8DA')

Je vydáno volání MQOPEN, MQPUTnebo MQPUT1 pro otevření fronty klastru nebo vložení zprávy do ní. Uživatelská procedura pracovní zátěže klastru definovaná atributem ClusterWorkloadExit správce front neočekávaně selhává nebo neodpovídá včas.

Do systémového protokolu v systému IBM MQ for z/OS se zapíše zpráva s dalšími informacemi o této chybě.

Následná 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 systému z/OSnelze načíst uživatelskou proceduru pracovní zátěže klastru.

Do systémového protokolu se zapíše zpráva a zpracování pokračuje, jako by byl atribut ClusterWorkloadExit prázdný.

Multi V systému Multiplatformsje pro připojení ke správci front vydáno volání MQCONN nebo MQCONNX . Volání selže, protože uživatelskou proceduru pracovní zátěže klastru definovanou atributem ClusterWorkloadExit správce front nelze načíst.

MQRC_CLUSTER_PUT_INHIBITED (2268 X'8DC')

Pro frontu klastru se vydá volání MQOPEN s účinnými volbami MQOO_OUTPUT a MQOO_BIND_ON_OPEN . Všechny instance fronty v klastru jsou aktuálně blokovány tím, že je atribut InhibitPut nastaven na hodnotu MQQA_PUT_INHIBITED. Protože nejsou k dispozici žádné instance fronty pro příjem zpráv, volání MQOPEN se nezdaří.

Tento kód příčiny se vyskytuje pouze v případě, že jsou splněny obě následující podmínky:

- Neexistuje žádná lokální instance fronty. Pokud existuje lokální instance, volání MQOPEN bude úspěšné, i když je lokální instance zablokována.
- Pro frontu neexistuje žádná uživatelská procedura pracovní zátěže klastru nebo existuje uživatelská procedura pracovní zátěže klastru, ale nezvolí instanci fronty. (Pokud uživatelská procedura pracovní zátěže klastru zvolí instanci fronty, volání MQOPEN bude úspěšné, i když je tato instance zablokována.)

Je-li ve volání MQOPEN zadána volba MQOO_BIND_NOT_FIXED , může být volání úspěšné i v případě, že jsou všechny fronty v klastru blokovány. Avšak následné volání MQPUT může selhat, pokud jsou všechny fronty v době volání stále zablokovány.

MQRC_CLUSTER_RESOLUTION_ERROR (2189 X'88D')

1. Je vydáno volání MQOPEN, MQPUTnebo MQPUT1 pro otevření fronty klastru nebo vložení zprávy do ní. Definici fronty nelze správně interpretovat, protože je vyžadována odezva od správce front úplného úložiště, ale žádná není k dispozici.
2. Pro objekt tématu s určením PUBSCOPE (ALL) nebo SUBSCOPE (ALL) je vydáno volání MQOPEN, MQPUT, MQPUT1 nebo MQSUB . Definici tématu klastru nelze správně interpretovat, protože je vyžadována odezva od správce front úplného úložiště, ale žádná není k dispozici.

MQRC_CLUSTER_RESOURCE_ERROR (2269 X'8DD')

Pro frontu klastru je vydáno volání MQOPEN, MQPUTnebo MQPUT1 . Při pokusu o použití prostředku požadovaného pro klastrování došlo k chybě.

MQRC_NO_DESTINATIONS_AVAILABLE (2270 X'8DE')

Je vydáno volání MQPUT nebo MQPUT1 pro vložení zprávy do fronty klastru. V době volání již nejsou v klastru žádné instance fronty. MQPUT selže a zpráva se neodešle.

K chybě může dojít, je-li ve volání MQOPEN , které otevře frontu, uveden parametr MQOO_BIND_NOT_FIXED , nebo je-li k vložení zprávy použit parametr MQPUT1 .

MQRC_STOPPED_BY_CLUSTER_EXIT (2188 X'88C')

Je vydáno volání MQOPEN, MQPUTnebo MQPUT1 pro otevření nebo vložení zprávy do fronty klastru. Uživatelská procedura pracovní zátěže klastru odmítne volání.

z/OS Používání a psaní aplikací na systému IBM MQ for z/OS

Aplikace IBM MQ for z/OS lze vytvořit z programů, které jsou spuštěny v mnoha různých prostředích. To znamená, že mohou využívat zařízení, která jsou k dispozici ve více než jednom prostředí.

Tyto informace vysvětlují prostředky systému IBM MQ , které jsou k dispozici pro programy spuštěné v jednotlivých podporovaných prostředích. Kromě toho

- Informace o použití IBM MQ-CICS bridgenaleznete v tématu Použití IBM MQ s produktem CICS .
- Informace o použití produktu IMS a mostu IMS viz “IMS a IMS přemostění aplikací na IBM MQ for z/OS” na stránce 66.

Pomocí následujících odkazů získáte další informace o používání a psaní aplikací na webu IBM MQ for z/OS:

- “Funkce IBM MQ for z/OS závislé na prostředí” na stránce 859
- “Prostředky ladění, podpora synchronizačního bodu a podpora zotavení” na stránce 860

- “Rozhraní IBM MQ for z/OS s prostředím aplikace” na stránce 861
- “Psaní aplikací z/OS UNIX System Services” na stránce 862
- “Programování aplikací se sdílenými frontami” na stránce 866

Související pojmy

“Přehled rozhraní fronty zpráv” na stránce 700
Informace o komponentách rozhraní MQI (Message Queue Interface).

“Připojení ke správci front a odpojení od něj” na stránce 713
Chcete-li používat programovací služby IBM MQ , musí mít program připojení ke správci front. Pomocí těchto informací se dozvíte, jak se připojit ke správci front a jak se od něj odpojit.

“Otevírání a zavírání objektů” na stránce 720
Tyto informace poskytují náhled na otevírání a zavírání objektů IBM MQ .

“Vkládání zpráv do fronty” na stránce 730
Pomocí těchto informací se dozvíte, jak vkládat zprávy do fronty.

“Získávání zpráv z fronty” na stránce 744
Pomocí těchto informací získáte informace o získávání zpráv z fronty.

“Zjišťování a nastavení atributů objektu” na stránce 822
Atributy jsou vlastnosti, které definují charakteristiky objektu IBM MQ .

“Potvrzení a zálohování jednotek práce” na stránce 825
Tyto informace popisují, jak potvrdit a vrátit zpět všechny obnovitelné operace get a put, které se vyskytly v transakci.

“Spuštění aplikací IBM MQ pomocí spouštěčů” na stránce 836
Získejte informace o spouštěčích a o tom, jak spouštět aplikace IBM MQ pomocí spouštěčů.

“Práce s rozhraním MQI a klastry” na stránce 854
Existují speciální volby pro volání a návratové kódy, které se týkají klastrování.

“IMS a IMS přemostění aplikací na IBM MQ for z/OS” na stránce 66
Tyto informace vám pomohou psát aplikace IMS pomocí IBM MQ.

z/OS *Funkce IBM MQ for z/OS závislé na prostředí*

Tyto informace použijte při zvažování funkcí IBM MQ for z/OS .

Hlavní rozdíly, které je třeba vzít v úvahu mezi funkcemi IBM MQ v prostředích, ve kterých je spuštěn produkt IBM MQ for z/OS , jsou:

- Produkt IBM MQ for z/OS dodává následující monitory spouštěčů:
  - CKTI pro použití v prostředí CICS
  - CSQQTRMN pro použití v prostředí IMS

  Chcete-li spouštět aplikace v jiných prostředích, musíte napsat vlastní modul.
- Synchronizace pomocí dvoufázového potvrzování je podporována v prostředích CICS a IMS . Je také podporován v dávkovém prostředí z/OS pomocí správy transakcí a služeb RRS (Obnovitelný správce prostředků). Jednofázové potvrzování je podporováno v prostředí z/OS samotným produktem IBM MQ .
- Pro dávková prostředí a prostředí IMS poskytuje rozhraní MQI volání pro připojení programů ke správci front a pro jejich odpojení od správce front. Programy se mohou připojit k více než jednomu správci front.
- Systém CICS se může připojit pouze k jednomu správci front. To se může stát, když se zahájí CICS , pokud je název subsystému definován v úloze spuštění systému CICS . Volání připojení a odpojení MQI jsou v prostředí CICS tolerována, ale nemají žádný vliv.
- Uživatelská procedura přechodu rozhraní API umožňuje programu zasahovat do zpracování všech volání MQI. Tato uživatelská procedura je k dispozici pouze v prostředí CICS .

- V produktu CICS na víceprocesorových systémech je získána určitá výkonová výhoda, protože volání MQI lze provádět pod více TCB produktu z/OS . Další informace viz příručka Plánování v z/OS *IBM MQ for z/OS Koncepty a plánování*.

Tyto funkce jsou shrnuty v části Tabulka 130 na stránce 860.

Tabulka 130. z/OS funkce prostředí

	CICS	**IMS**	**Dávka/TSO**
Monitor spouštěčů byl dodán	Ano	Ano	Ne
Dvoufázové potvrzení	Ano	Ano	Ano
jednofázové potvrzení	Ano	Ne	Ano
Připojení/odpojení volání MQI	Tolerováno	Ano	Ano
Uživatelská procedura napříč rozhraním API	Ano	Ne	Ne

Poznámka: Dvoufázové potvrzování je podporováno v prostředí Batch/TSO pomocí RRS.

z/OS *Prostředky ladění, podpora synchronizačního bodu a podpora zotavení*

Pomocí těchto informací získáte informace o ladicích zařízeních programu, podpoře synchronizačního bodu a podpoře obnovy.

Prostředky ladění programu

Produkt IBM MQ for z/OS poskytuje prostředek trasování, který můžete použít k ladění programů ve všech prostředích.

Kromě toho můžete v prostředí CICS použít:

- Nástroj CICS pro diagnostiku provádění (CEDF)
- Řídicí transakce trasování CICS (CETR)
- Uživatelská procedura rozhraní API IBM MQ for z/OS

Na platformě z/OS můžete použít libovolný dostupný interaktivní nástroj pro ladění, který je podporován programovacím jazykem, který používáte.

Podpora synchronizačních bodů

Synchronizace začátku a konce jednotek práce je nezbytná v prostředí pro zpracování transakcí, aby bylo možné bezpečně používat zpracování transakcí.

To je plně podporováno produktem IBM MQ for z/OS v prostředích CICS a IMS . Úplná podpora znamená spolupráci mezi správci prostředků, takže jednotky práce mohou být potvrzeny nebo odvolány jednotně pod kontrolou CICS nebo IMS. Příklady správců prostředků jsou Db2, CICS Řízení souborů, IMSa IBM MQ for z/OS.

z/OS dávkové aplikace mohou používat volání IBM MQ for z/OS k poskytnutí prostředku jednofázového potvrzování. To znamená, že sadu operací fronty definovanou aplikací lze potvrdit nebo zálohovat bez odkazu na jiné správce prostředků.

Dvoufázové potvrzování je také podporováno v dávkovém prostředí z/OS pomocí služeb správy transakcí a obnovitelných služeb správce prostředků (RRS). Další informace viz Syncpoints in z/OS dávkové aplikace.

Podpora obnovy

Pokud je během transakce přerušeno připojení mezi správcem front a systémem CICS nebo IMS , některé jednotky práce nemusí být úspěšně vráceny zpět.

Tyto jednotky práce jsou však vyřešeny správcem front (pod kontrolou správce synchronizačních bodů) při opětovném vytvoření připojení k systému CICS nebo IMS .

z/OS *Rozhraní IBM MQ for z/OS s prostředím aplikace*

Chcete-li povolit aplikacím běžícím v různých prostředích odesílat a přijímat zprávy prostřednictvím sítě s frontami zpráv, produkt IBM MQ for z/OS poskytuje *adaptér* pro každé podporované prostředí.

Tyto adaptéry jsou rozhraním mezi aplikačními programy a subsystémy IBM MQ for z/OS . Umožňují programům používat rozhraní MQI.

z/OS *Dávkový adaptér*

Pomocí těchto informací se dozvíte o dávkovém adaptéru a protokolu potvrzení, který podporuje.

Dávkový adaptér poskytuje přístup k prostředkům IBM MQ for z/OS pro programy spuštěné v:

- Režim úlohy (TCB)
- Stav problému nebo supervizora
- Režim řízení primárního adresního prostoru

Programy nesmí být v režimu křížové paměti.

Připojení mezi aplikačními programy a produktem 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 v prostředcích vlastněných společností IBM MQ for z/OS ; nepodporuje protokoly vícefázového potvrzování.

z/OS *Dávkový adaptér RRS*

Pomocí těchto informací získáte informace o dávkovém adaptéru RRS a dvou dávkových adaptérech RRS poskytovaných produktem IBM MQ.

Adaptér správy transakcí a služeb RRS (Obnovitelné služby správce prostředků):

- Používá z/OS RRS pro řízení potvrzení.
- Podporuje simultánní připojení k více subsystémům IBM MQ spuštěných na jedné instanci z/OS z jedné 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-uložené procedury prováděné v adresním prostoru Db2-uložené procedury, který je spravován správcem pracovní zátěže (WLM) v systému z/OS.
- Podporuje schopnost přepínat dávkový podproces IBM MQ mezi TCB.

Produkt IBM MQ for z/OS poskytuje dva adaptéry dávky RRS:

CSQBRSTB
Tento adaptér vyžaduje změnu libovolného příkazu MQCMIT na SRRCMIT a libovolného příkazu MQBACK na SRRBACK ve vaší aplikaci IBM MQ . (Pokud kódujete MQCMIT nebo MQBACK v aplikaci propojené s CSQBRSTB, obdržíte zprávu MQRC_ENVIRONMENT_ERROR.)

CSQBRRSI
Tento adaptér umožňuje aplikaci IBM MQ používat buď MQCMIT a MQBACK, nebo SRRCMIT a SRRBACK.

Poznámka: CSQBRSTB a CSQBRRSI jsou dodávány s atributy sestavení AMODE (31) RMODE (ANY). Pokud vaše aplikace načte jeden ze stubů pod řádkem 16 MB, nejprve tento stub znovu propojte s RMODE (24).

Migration

Existující aplikace Batch/TSO IBM MQ můžete migrovat tak, aby používaly koordinaci RRS s několika nebo žádnými změnami.

Pokud aplikaci IBM MQ propojíte s adaptérem CSQBRRSI, MQCMIT a MQBACK synchronizují vaši pracovní jednotku v produktu IBM MQ a ve všech ostatních správcích prostředků s podporou RRS. Pokud aplikaci IBM MQ propojíte s adaptérem CSQBRSTB, změňte hodnotu MQCMIT na SRRCMIT a hodnotu MQBACK na SRRBACK. Druhý přístup je vhodnější; jasně označuje, že synchronizační bod není omezen pouze na prostředky IBM MQ .

z/OS *Adaptér IMS*

Používáte-li adaptér IMS ze systému IBM MQ for z/OS , ujistěte se, že produkt IMS může získat dostatek úložiště pro uložení zpráv o délce až 100 MB.

Poznámka pro uživatele

IMS Adaptér poskytuje přístup k prostředkům IBM MQ for z/OS pro:

- Online programy pro zpracování zpráv (MPP)
- Interaktivní programy rychlé cesty (IFP)
- Programy pro zpracování dávkových zpráv (BMP)

Chcete-li tyto prostředky použít, musí být programy spuštěny v režimu úlohy (TCB) a v problémovém stavu; nesmí být v režimu křížové paměti nebo v režimu registrace přístupu.

Adaptér poskytuje podproces připojení z řídicího bloku úlohy aplikace (TCB) do produktu IBM MQ. Adaptér podporuje protokol dvoufázového potvrzování pro změny provedené v prostředcích vlastněných produktem IBM MQ for z/OS, přičemž IMS vystupuje jako koordinátor synchronizačního bodu.

Adaptér také poskytuje program pro monitorování spouštěčů, který může spouštět programy automaticky při splnění určitých podmínek spouštěče ve frontě. Další informace viz téma “Spuštění aplikací IBM MQ pomocí spouštěčů” na stránce 836.

Pokud píšete dávkové programy DL/I, postupujte podle pokynů uvedených v tomto tématu pro dávkové programy z/OS .

z/OS *Psaní aplikací z/OS UNIX System Services*

Dávkový adaptér podporuje připojení správce front z dávkového adresního prostoru a adresního prostoru TSO.

V případě adresního 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 daném okamžiku pouze jednu instanci připojení ke konkrétnímu správci front).
- Ke stejnému správci front se může připojit více TCB (ale manipulátor správce front vrácený v libovolném volání MQCONN nebo MQCONNX je svázán s vydávajícím TCB a nemůže být používán žádným jiným TCB).

z/OS UNIX System Services podporuje dva typy volání pthread_create:

1. Těžké podprocesy, spusťte jeden pro každý blok TCB, které jsou ATTACHed a DETACHed na začátku a konci podprocesu z/OS.
2. Středně těžké podprocesy, spouštějte jeden pro každý blok TCB, ale blok TCB může být jedním z fondu přerušitelných centrálních bank. Aplikace musí provést veškeré potřebné vyčištění aplikace, protože pokud je připojena k serveru, výchozí ukončení podprocesu, které může poskytnout server při ukončení úlohy (TCB), **není** vždy řízeno.

Odlehčené podprocesy nejsou podporovány. (Pokud aplikace vytvoří trvalé podprocesy, které odbavují své vlastní pracovní požadavky, je ***aplikace*** zodpovědná za vyčištění všech prostředků před spuštěním dalšího pracovního požadavku.)

IBM MQ for z/OS podporuje z/OS UNIX System Services podprocesy používající dávkový adaptér:

1. Vlákna s vysokou hmotností jsou plně podporována jako dávková připojení. Každý podproces běží ve svém vlastním TCB, který je připojen a odpojen při spuštění a ukončení podprocesu. Pokud podproces skončí před vyvoláním volání MQDISC, produkt IBM MQ for z/OS provede standardní vyčištění úlohy, což zahrnuje potvrzení všech nevyřízených pracovních jednotek, pokud byl podproces ukončen normálně, nebo jeho zrušení, pokud byl podproces nestandardně ukončen.
2. Podprocesy střední váhy jsou plně podporovány, ale pokud má být blok TCB znovu použit jiným podprocesem, musí aplikace zajistit, aby bylo před příštím spuštěním podprocesu vydáno volání MQDISC, kterému předchází MQCMIT nebo MQBACK. To znamená, že pokud aplikace zřídila obslužnou rutinu přerušení programu a aplikace se pak neukončí, obslužná rutina přerušení musí před opětovným použitím TCB pro jiný podproces zadat volání MQCMIT a MQDISC.

Poznámka: Modely podprocesů **nepodporují** přístup ke společným prostředkům IBM MQ z více podprocesů.

z/OS *Uživatelská procedura přechodu rozhraní API pro z/OS*

Toto téma obsahuje informace o programovacím rozhraní, které jsou citlivé na produkt.

Uživatelská procedura je bod v kódu dodaném IBM, kde můžete spustit svůj vlastní kód. Produkt IBM MQ for z/OS poskytuje *uživatelskou proceduru přechodu rozhraní API* , kterou můžete použít k zachycení volání MQI a k monitorování nebo úpravě funkce volání MQI. Tento oddíl popisuje, jak používat uživatelskou proceduru pro přechod rozhraní API, a popisuje ukázkový uživatelský program dodávaný s produktem IBM MQ for z/OS.

Tento oddíl je použitelný pouze pro uživatele produktu CICS TS V3.1 a starší. Uživatelé produktu CICS TS V3.2 a novější by se měli obrátit na sekci CICS Integrace s produktem IBM MQ v dokumentaci k produktu CICS .

Poznámka

Uživatelská procedura přechodu rozhraní API je vyvolána pouze adaptérem CICS produktu IBM MQ for z/OS. Uživatelský program je spuštěn v adresním prostoru CICS .

z/OS *Psaní vlastního uživatelského programu*

Můžete použít ukázkový uživatelský program CSQCAPX (API-crossing exit program), který je dodáván s produktem IBM MQ for z/OS jako rámec pro váš vlastní program.

To je popsáno v tématu “Ukázkový uživatelský program překračující rozhraní API, CSQCAPX” na stránce 864.

Chcete-li při psaní uživatelského programu vyhledat název volání MQI vydaného aplikací, zkontrolujte pole *ExitCommand* struktury MQXP. Chcete-li vyhledat počet parametrů volání, zkontrolujte pole *ExitParmCount* . Pomocí 16bajtového pole *ExitUserArea* můžete uložit adresu libovolného dynamického úložiště, které aplikace získá. Toto pole je zachováno při všech voláních uživatelské procedury a má stejnou dobu životnosti jako úloha CICS .

Používáte-li CICS Transakční server V3.2, musíte napsat uživatelský program, aby zajišťoval neporušenost vláken, a deklarovat svůj uživatelský program jako zabezpečen neporušenost vláken. Používáte-li dřívější vydání produktu CICS , doporučuje se také zapsat a deklarovat své uživatelské programy jako bezpečné pro podprocesy, aby byly připraveny k migraci na server CICS Transaction Server V3.2.

Váš uživatelský program může potlačit provádění volání MQI vrácením příkazu MQXCC_SUPPRESS_FUNCTION nebo MQXCC_SKIP_FUNCTION v poli *ExitResponse* . Chcete-li povolit provedení volání (a opětné vyvolání uživatelského programu po dokončení volání), musí uživatelský program vrátit hodnotu MQXCC_OK.

Při vyvolání po volání MQI může uživatelský program zkontrolovat a upravit kódy dokončení a příčiny nastavené voláním.

Poznámky k použití

Zde jsou některé obecné body, které je třeba zvážit při psaní svého uživatelského programu:

- Z výkonnostních důvodů napište svůj program v assembleru. Pokud jej napíšete v některém z ostatních jazyků podporovaných produktem IBM MQ for z/OS, musíte poskytnout vlastní soubor definice dat.
- Linkování programu jako AMODE (31) a RMODE (ANY).
- Chcete-li definovat blok parametrů ukončení pro váš program, použijte makro v jazyce assembleru, CMQXPA.
- Zadejte CONCURRENCY (THREADSAFE), když definujete uživatelský program a všechny programy, které váš uživatelský program volá.
- Používáte-li funkci ochrany úložiště CICS Transaction Server for z/OS , musí být váš program spuštěn v prováděcím klíči CICS . To znamená, že musíte zadat EXECKEY (CICS) při definování uživatelského programu i programů, kterým předává řízení. Informace o uživatelských programech CICS a prostředku ochrany úložiště 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 CICS . Může také použít libovolné volání MQI s výjimkou volání MQCONN, MQCONNX a MQDISC. Avšak žádná volání MQI v rámci uživatelského programu nevyvolává uživatelský program podruhé.
- Váš program může vydat příkazy EXEC CICS SYNCPOINT nebo EXEC CICS SYNCPOINT ROLLBACK. Tyto příkazy však potvrdí nebo odvolají ***všechny*** aktualizace provedené úlohou až do okamžiku, kdy byla uživatelská procedura použita, a jejich použití se proto nedoporučuje.
- Váš program musí skončit zadáním příkazu EXEC CICS RETURN. Nesmí přenášet řízení pomocí příkazu XCTL.
- Uživatelské procedury se zapisují jako rozšíření kódu IBM MQ for z/OS . Ujistěte se, že uživatelská procedura nenarušuje žádné programy nebo transakce IBM MQ for z/OS , které používají rozhraní MQI. Ty jsou obvykle označeny předponou CSQ nebo CK.
- Pokud je CSQCAPX definován jako CICS, systém CICS se pokusí načíst uživatelský program, když se CICS připojí k IBM MQ for z/OS. Pokud je tento pokus úspěšný, odešle se zpráva CSQC301I na panel CKQC nebo na systémovou konzolu. Pokud je načtení neúspěšné (například pokud zaváděcí modul neexistuje v žádné z knihoven ve zřetězení DFHRPL), odešle se zpráva CSQC315 na panel CKQC nebo na systémovou konzolu.
- Protože parametry v komunikační oblasti jsou adresy, musí být uživatelský program definován jako lokální pro systém CICS (tj. ne jako vzdálený program).

z/OS *Ukázkový uživatelský program překračující rozhraní API, CSQCAPX*

Ukázkový uživatelský program je dodáván jako program v jazyku asembleru. Zdrojový soubor (CSQCAPX) je dodáván v knihovně **thlqual**.SCSQASMS (kde **thlqual** je kvalifikátor vyšší úrovně používaný vaší instalací). Tento zdrojový soubor obsahuje pseudokód, který popisuje logiku programu.

Ukázkový program obsahuje inicializační kód a rozvržení, které můžete použít při psaní vlastních uživatelských programů.

Ukázka ukazuje, jak:

- Nastavit blok parametrů ukončení
- Adresovat bloky parametrů volání a ukončení
- Určit, pro které volání MQI je volána uživatelská procedura
- Určete, zda je uživatelská procedura vyvolávána před nebo po zpracování volání MQI.
- Vložit zprávu do fronty dočasného úložiště CICS
- Použijte makro DFHEIENT pro získání dynamického úložiště, abyste zachovali reentrancy

- Použít DFHEIBLK pro řídicí blok rozhraní exec CICS
- Chybové podmínky depeše
- Vrátit řízení volajícímu

Návrh ukázkového uživatelského programu

Ukázkový uživatelský program zapisuje zprávy do CICS fronty dočasného úložiště (CSQ1EXIT), aby zobrazil operaci uživatelské procedury.

Zprávy ukazují, zda je uživatelská procedura vyvolávána před nebo po volání MQI. Je-li uživatelská procedura vyvolána po volání, zpráva obsahuje kód dokončení a kód příčiny vrácený voláním. Ukázka používá pojmenované konstanty z makra CMQXPA ke kontrole typu záznamu (tj. před nebo po volání).

Ukázka neprovádí žádnou funkci monitorování, ale jednoduše umístí zprávy s časovým razítkem do fronty CICS označující typ volání, které zpracovává. Tato volba poskytuje informace o výkonu rozhraní MQI a o správné funkci uživatelského programu.

Poznámka: Ukázkový uživatelský program vydá šest volání EXEC CICS pro každé volání MQI provedené za běhu programu. Pokud použijete tento uživatelský program, výkon produktu IBM MQ for z/OS se sníží.

z/OS *Příprava a použití křížové uživatelské procedury rozhraní API*

Ukázková uživatelská procedura je dodávána pouze ve zdrojovém formátu.

Chcete-li použít ukázkovou uživatelskou proceduru nebo uživatelský program, který jste napsali, vytvořte zaváděcí knihovnu, jako byste ji používali pro jakýkoli jiný program CICS , jak je popsáno v tématu “Sestavení aplikací CICS v produktu z/OS” na stránce 993.

- V případě serveru CICS Transaction Server pro z/OS a CICS pro MVS/ESAplatí, že když aktualizujete datovou sadu definice systému CICS (CSD), potřebné definice jsou ve členu **thlqual**.SCSQPROC (CSQ4B100).

  Poznámka: Definice používají příponu MQ. Pokud je tato přípona již ve vašem podniku použita, je třeba ji změnit před fází sestavení.

Pokud použijete výchozí dodané definice programu CICS , uživatelský program CSQCAPX se nainstaluje ve stavu ***zakázáno*** . Důvodem je, že použití uživatelského programu může vést k významnému snížení výkonu.

Chcete-li dočasně aktivovat uživatelskou proceduru přechodu rozhraní API, postupujte takto:

1. Zadejte příkaz **CEMT S PROGRAM(CSQCAPX) ENABLED** z hlavního terminálu CICS .
2. Spusťte transakci CKQC a použijte volbu 3 ve stahovací nabídce Připojení, abyste změnili stav uživatelské procedury křížení rozhraní API na **Povoleno**.

Chcete-li spustit produkt IBM MQ for z/OS s trvale povolenou uživatelskou procedurou pro přechod rozhraní API s volbou CICS Transakční server pro z/OS a CICS pro MVS/ESA, proveďte jednu z následujících akcí:

- Změňte definici CSQCAPX ve členu CSQ4B100změnou STATUS (DISABLED) na STATUS (ENABLED). Definici CSD CICS můžete aktualizovat pomocí dávkového programu DFHCSDUP dodaného společností CICS.
- Změňte definici CSQCAPX ve skupině CSQCAT1 změnou stavu z DISABLED na ENABLED.

V obou případech musíte skupinu přeinstalovat. To můžete provést studeným spuštěním systému CICS nebo pomocí transakce CICS CEDA pro přeinstalaci skupiny za běhu CICS .

Poznámka: Použití CEDA může způsobit chybu, pokud jsou některé položky ve skupině momentálně používány.

Konec informací o programovacím rozhraní citlivému na produkt.

z/OS *Programování aplikací se sdílenými frontami*

Toto téma obsahuje 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 se sdílenými frontami.

z/OS *Serializace aplikací*

Určité typy aplikací mohou vyžadovat, aby se zprávy načítaly z fronty přesně ve stejném pořadí, v jakém byly přijaty do fronty.

Pokud se například produkt IBM MQ používá ke stínování aktualizací databáze na vzdáleném systému, musí být zpráva popisující aktualizaci záznamu zpracována po zprávě popisující vložení tohoto záznamu. V lokálním prostředí front je toho často dosahováno aplikací, která získává zprávy otevírající frontu s volbou MQOO_INPUT_EXCLUSIVE, čímž brání jakékoli jiné aplikaci, která získává 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 pro tabulku A mají identifikátor korelace A a pro tabulku B identifikátor korelace B) a aplikace chtějí získat zprávy pro aktualizace tabulky A a aktualizace tabulky B souběžně, jednoduchý mechanizmus výlučného otevření fronty není možný.

Má-li tento typ aplikace využívat vysokou dostupnost sdílených front, můžete se rozhodnout, že v případě selhání primární aplikace nebo správce front by měla převzít jiná instance aplikace, která přistupuje ke stejným sdíleným frontám spuštěným v sekundárním správci front.

Pokud primární správce front selže, dojde ke dvěma věcem:

- Obnova typu peer sdílené fronty zajišťuje, že veškeré neúplné aktualizace z primární aplikace jsou dokončeny nebo vráceny zpět.
- Sekundární aplikace převezme zpracování fronty.

Sekundární aplikace se může spustit před vyřešením všech nedokončených pracovních jednotek, což může vést k tomu, že sekundární aplikace načte zprávy mimo pořadí. Chcete-li tento typ problému vyřešit, může se aplikace rozhodnout, že bude *serializovanou aplikací*.

Serializovaná aplikace používá volání MQCONNX pro připojení ke správci front a určuje značku připojení při připojení, která je pro danou aplikaci jedinečná. Všechny jednotky práce provedené aplikací jsou označeny značkou připojení. Produkt IBM MQ zajišťuje serializaci pracovních jednotek v rámci skupiny sdílení front se stejnou značkou připojení (podle voleb serializace ve volání MQCONNX).

To znamená, že pokud primární aplikace používá volání MQCONNX se značkou připojení `Database shadow retrievera` sekundární aplikace převzetí se pokusí použít volání MQCONNX s identickou značkou připojení, sekundární aplikace se nemůže připojit k druhému IBM MQ , dokud nebudou dokončeny všechny nevyřízené primární pracovní jednotky, v tomto případě prostřednictvím zotavení typu peer.

Zvažte použití techniky serializované aplikace pro aplikace, které závisí na přesné posloupnosti zpráv ve frontě. Konkrétně se jedná o následující podmínky:

- Aplikace, které nesmí být restartovány po selhání aplikace nebo správce front, dokud nebudou dokončeny všechny operace potvrzení a vrácení pro předchozí provedení aplikace.

  V tomto případě je technika serializované aplikace použitelná pouze v případě, že aplikace pracuje v synchronizačním bodu.
- Aplikace, které se nesmí spustit, když je již spuštěna jiná instance stejné aplikace.

  V tomto případě je technika serializované 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.

z/OS *Aplikace, které nejsou vhodné pro použití se sdílenými frontami*

Některé funkce produktu IBM MQ nejsou podporovány, když používáte sdílené fronty, takže aplikace, které používají tyto funkce, nejsou vhodné pro prostředí sdílených front.

Při návrhu aplikací se sdílenou frontou zvažte následující body:

- Indexování fronty je omezeno pro sdílené fronty. Chcete-li použít identifikátor zprávy nebo identifikátor korelace k výběru zprávy, kterou chcete získat 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 pouze podle identifikátoru korelace, musí mít fronta typ indexu MQIT_CORREL_ID.
- Jako sdílené fronty nelze používat dočasné dynamické fronty. Můžete však použít trvalé dynamické fronty. Modely pro sdílené dynamické fronty mají hodnotu DEFTYPE SHAREDYN (sdílená dynamická), i když jsou vytvořeny a zničeny stejným způsobem jako PERMDYN (trvalé dynamické) fronty.

z/OS *Rozhodování, zda sdílet neaplikační fronty*

Tyto informace použijte při zvažování sdílení neaplikačních front.

Existují jiné fronty než fronty aplikací, které byste mohli zvážit sdílení:

Inicializační fronty

Pokud definujete sdílenou inicializační frontu, nemusíte mít spuštěný monitor spouštěčů v každém správci front ve skupině sdílení front, pokud je spuštěn alespoň jeden monitor spouštěčů. (Sdílenou inicializační frontu můžete použít i v případě, že je v každém správci front ve skupině sdílení front spuštěn monitor spouštěčů.)

Máte-li sdílenou frontu aplikací a používáte-li typ spouštěče EVERY (nebo typ spouštěče FIRST s malým intervalem spouštěče, který se chová jako typ spouštěče EVERY), musí být inicializační fronta vždy sdílenou frontou. Další informace o tom, kdy použít sdílenou inicializační frontu, viz Tabulka 131 na stránce 868.

SYSTÉM.* fronty

Můžete definovat SYSTEM.ADMIN.* fronty používané k uchování zpráv událostí jako sdílených front. To může být užitečné při kontrole vyvažování zátěže, pokud dojde k výjimce. Každá zpráva události vytvořená produktem IBM MQ obsahuje identifikátor korelace označující, který správce front ji vytvořil.

Musíte definovat SYSTEM.QSG.* fronty používané pro sdílené kanály a fronty v rámci skupiny jako sdílené fronty.

Můžete také změnit definice SYSTEM.DEFAULT.LOCAL.QUEUE , která má být sdílena, nebo definujte vlastní výchozí definici sdílené fronty. Další informace viz Definování systémových objektů pro IBM MQ for z/OS .

Nemůžete definovat žádný jiný SYSTEM.* fronty jako sdílené fronty.

z/OS *Migrace existujících aplikací pro použití sdílených front*

Kódy příčiny, spouštění a volání rozhraní API MQINQ mohou ve sdíleném prostředí fronty pracovat odlišně.

Informace o migraci existujících front do sdílených front naleznete v tématu Migrace nesdílených front do sdílených front .

Při migraci existujících aplikací zvažte následující skutečnosti, které mohou v prostředí sdílených front fungovat jiným způsobem:

Kódy příčin

Při migraci existujících aplikací tak, aby používaly sdílené fronty, zkontrolujte nové kódy příčiny, které lze vydat.

Spouątění

Používáte-li sdílenou frontu aplikací, spouštění pracuje pouze na potvrzených zprávách (ve frontě nesdílených aplikací, spouštění pracuje na všech zprávách).

Pokud ke spuštění aplikací použijete spouštěče, možná budete chtít použít sdílenou inicializační frontu. Tabulka 131 na stránce 868 popisuje, co je třeba zvážit při rozhodování, který typ inicializační fronty použít.

Tabulka 131. Kdy použít sdílenou inicializační frontu

	Nesdílená fronta aplikací	**Sdílená fronta aplikací**
Nesdílená inicializační fronta	Stejně jako u předchozích verzí.	Pokud použijete typ spouštěče FIRST nebo DEPTH, můžete použít nesdílenou inicializační frontu se sdílenou frontou aplikací. Mohou být generovány další zprávy spouštěče, ale toto nastavení je vhodné pro spouštění přerušitelných aplikací (jako např. CICS bridge) a poskytuje vysokou dostupnost. Pro typ spouštěče FIRST nebo DEPTH zpráva spouštěče spustí instanci aplikace v každém správci front, který spouští monitor spouštěčů a který dosud nemá otevřenou frontu aplikací pro vstup. Pro každého správce front je vygenerována jedna zpráva spouštěče. Pokud je pro nesdílenou lokální inicializační frontu spuštěn více než jeden monitor spouštěčů v konkrétním správci front, budou soupeřit o zpracování zprávy.
Sdílená inicializační fronta	Nepoužívejte sdílenou inicializační frontu s nesdílenou frontou aplikací.	V případě typu spouštěče EVERY, když aplikace vloží zprávu do sdílené fronty aplikací, správce front vložení určí, kteří správci front mají zájem o spouštěč-každou událost a odešle oznámení jednomu z těchto správců front. V oznámeném správci front je výslednou akcí vygenerování zprávy spouštěče do inicializační fronty. **Poznámka:** Máte-li sdílenou frontu aplikací, která má typ spouštěče KAŽDÝ, použijte sdílenou inicializační frontu, nebo můžete za určitých okolností ztratit zprávy spouštěče, například selhání správce front. Pro typ spouštěče FIRST nebo DEPTH je pro každého správce front, který má uvedenou inicializační frontu otevřenou pro vstup, vygenerována jedna zpráva spouštěče. **Poznámka:** Pro typ spouštěče FIRST nebo DEPTH, je-li jedna instance monitoru spouštěčů zaneprázdněná, ponechá se tak možnost, že méně vytížené monitory spouštěčů zpracují více než jednu zprávu spouštěče ze sdílené inicializační fronty. Proto může být pro daného správce front spuštěno více instancí serverové aplikace. Všimněte si, že tyto vícenásobné instance jsou spuštěny jako výsledek zpracování více zpráv spouštěče. V případě typu spouštěče FIRST nebo DEPTH, pokud instance aplikace již obsluhuje frontu aplikací, nebude správce front, ke kterému je aplikace připojena, generovat jinou zprávu spouštěče.

MQINQ

Při použití volání MQINQ k zobrazení informací o sdílené frontě se hodnoty počtu volání MQOPEN , která mají otevřenou frontu pro vstup a výstup, vztahují pouze na správce front, který volání vydal. Nejsou

vytvořeny žádné informace o jiných správcích front ve skupině sdílení front, kteří mají otevřenou frontu.

z/OS IMS a IMS přemostění aplikací na IBM MQ for z/OS

Tyto informace vám pomohou psát aplikace IMS pomocí IBM MQ.

- Chcete-li používat synchronizační body a volání MQI v aplikacích IMS , viz “Psaní aplikací IMS pomocí IBM MQ” na stránce 66.
- Chcete-li psát aplikace, které používají most IBM MQ - IMS , viz “Zápis aplikací mostu IMS” na stránce 70.

Další informace o aplikacích mostu IMS a IMS v systému IBM MQ for z/OSzískáte pomocí následujících odkazů:

- “Psaní aplikací IMS pomocí IBM MQ” na stránce 66
- “Zápis aplikací mostu IMS” na stránce 70

Související pojmy

“Přehled rozhraní fronty zpráv” na stránce 700
Informace o komponentách rozhraní MQI (Message Queue Interface).

“Připojení ke správci front a odpojení od něj” na stránce 713
Chcete-li používat programovací služby IBM MQ , musí mít program připojení ke správci front. Pomocí těchto informací se dozvíte, jak se připojit ke správci front a jak se od něj odpojit.

“Otevírání a zavírání objektů” na stránce 720
Tyto informace poskytují náhled na otevírání a zavírání objektů IBM MQ .

“Vkládání zpráv do fronty” na stránce 730
Pomocí těchto informací se dozvíte, jak vkládat zprávy do fronty.

“Získávání zpráv z fronty” na stránce 744
Pomocí těchto informací získáte informace o získávání zpráv z fronty.

“Zjišťování a nastavení atributů objektu” na stránce 822
Atributy jsou vlastnosti, které definují charakteristiky objektu IBM MQ .

“Potvrzení a zálohování jednotek práce” na stránce 825
Tyto informace popisují, jak potvrdit a vrátit zpět všechny obnovitelné operace get a put, které se vyskytly v transakci.

“Spuštění aplikací IBM MQ pomocí spouštěčů” na stránce 836
Získejte informace o spouštěčích a o tom, jak spouštět aplikace IBM MQ pomocí spouštěčů.

“Práce s rozhraním MQI a klastry” na stránce 854
Existují speciální volby pro volání a návratové kódy, které se týkají klastrování.

“Používání a psaní aplikací na systému IBM MQ for z/OS” na stránce 858
Aplikace IBM MQ for z/OS lze vytvořit z programů, které jsou spuštěny v mnoha různých prostředích. To znamená, že mohou využívat zařízení, která jsou k dispozici ve více než jednom prostředí.

z/OS *Psaní aplikací IMS pomocí IBM MQ*

Při používání produktu IBM MQ v aplikacích IMS je třeba vzít v úvahu další aspekty. Patří mezi ně volání rozhraní API produktu MQ a mechanizmus používaný pro synchronizační bod.

Chcete-li se dozvědět více o psaní aplikací IMS na webu IBM MQ for z/OS, použijte následující odkazy:

- “Synchronizační body v aplikacích IMS” na stránce 67
- “Volání MQI v aplikacích IMS” na stránce 67

Omezení

Existují omezení, která mohou volání rozhraní API produktu IBM MQ používat aplikace používající adaptér IMS .

Následující volání rozhraní API IBM MQ nejsou podporována v rámci aplikace používající adaptér IMS :

- MQCB
- MQCB_FUNCTION
- MQCTL

Související pojmy

“Zápis aplikací mostu IMS” na stránce 70
Toto téma obsahuje informace o zápisu aplikací pro použití mostu IBM MQ - IMS .

z/OS *Synchronizační body v aplikacích IMS*

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

Chcete-li vrátit zpět všechny změny od předchozího kontrolního bodu, můžete použít volání IMS ROLB (odvolání). Další informace viz Volání ROLB v dokumentaci k produktu IMS .

Správce front je účastníkem protokolu dvoufázového potvrzování; koordinátorem je správce synchronizačních bodů IMS .

Všechny otevřené popisovače jsou uzavřeny adaptérem IMS v synchronizačním bodu (s výjimkou prostředí BMP s dávkovým zpracováním nebo bez použití zpráv). Důvodem je skutečnost, že při provádění volání MQCONN, MQCONNX a MQOPEN může jiný uživatel zahájit další transakci a při provádění volání MQPUT nebo MQGET je provedena kontrola zabezpečení produktu IBM MQ .

Avšak v prostředí WFI (Wait-for-Input) nebo PWFI (pseudo-Wait-for-Input) IMS neoznámí IBM MQ zavření popisovačů, dokud nedorazí další zpráva nebo dokud se aplikaci nevrátí stavový kód QC. Pokud aplikace čeká v oblasti IMS a některý z těchto manipulátorů patří do spouštěných front, nedojde ke spuštění, protože jsou fronty otevřené. Z tohoto důvodu by aplikace spuštěné v prostředí WFI nebo PWFI měly explicitně MQCLOSE manipulovat s frontou před provedením GU pro IOPCB pro další zprávu.

Pokud aplikace IMS (buď BMP, nebo MPP) vydá volání MQDISC, otevřené fronty se zavřou, ale neprovede se žádný implicitní synchronizační bod. Pokud aplikace skončí normálně, všechny otevřené fronty se zavřou a dojde k implicitnímu potvrzení. Pokud aplikace skončí abnormálně, všechny otevřené fronty se zavřou a dojde k implicitnímu vrácení.

z/OS *Volání MQI v aplikacích IMS*

Pomocí těchto informací získáte informace o používání volání MQI v aplikacích Server a v aplikacích Poptávka.

Tento oddíl se zabývá použitím volání MQI v následujících typech aplikací IMS :

- “Serverové aplikace” na stránce 870
- “Žádosti o dotazování” na stránce 873

Serverové aplikace

Zde je uveden přehled 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 produktu BMP používajícího tento model v produktu C/370. Program nejprve naváže komunikaci s produktem IMS a poté s produktem 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 produktu IMS určuje, zda byl program volán jako objekt BMP řízený zprávami nebo dávkově orientovaný, a řídí odpovídajícím způsobem připojení a obslužné rutiny front produktu IBM MQ :

```

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 produktu IBM MQ se připojí ke správci front a otevře fronty. V BMP řízeném zprávami je toto voláno po každém synchronizačním bodu IMS ; v dávkově orientovaném BMP je toto voláno pouze při spuštění 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 jednu pracovní jednotku pro každé vyvolání. Důvodem je skutečnost, že při použití synchronizačního bodu (GU) jsou uzavřeny manipulátory připojení a fronty a je doručena další zpráva produktu IMS . Toto omezení lze částečně překonat jedním z následujících způsobů:

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

  Jedná se o:

  – Čtení zprávy
  – Zpracování požadovaných aktualizací
  – Vložení odpovědi

  ve smyčce, dokud nebudou zpracovány všechny zprávy nebo dokud nebude zpracován nastavený maximální počet zpráv, při kterém se provede synchronizační bod.

  Tímto způsobem lze přistupovat pouze k určitým typům aplikací (například k jednoduché aktualizaci databáze nebo dotazu). Ačkoli lze zprávy odpovědí rozhraní MQI vložit s oprávněním původce zpracovávané zprávy rozhraní MQI, je třeba pečlivě řešit dopady na zabezpečení všech aktualizací prostředků produktu IMS .
- **Zpracování jedné zprávy na vyvolání MPP a zajištění vícenásobného plánování MPP pro zpracování všech dostupných zpráv.**

  Pomocí programu pro monitorování spouštěčů IBM MQ IMS (CSQQTRMN) naplánujte transakci MPP, když jsou ve frontě IBM MQ zprávy a žádné aplikace ji neobsluhují.

  Pokud monitor spouštěčů spustí protokol MPP, předá název správce front a název fronty programu, jak ukazuje následující extrakce kódu v jazyce 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, u kterého se očekává, že se jedná o dlouhotrvající úlohu, je lépe podporován v oblasti dávkového zpracování, ačkoli BMP nelze spustit pomocí CSQQTRMN.

Žádosti o dotazování

Typická aplikace systému IBM MQ , která zahajuje dotaz nebo aktualizaci, funguje takto:

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

Vzhledem k tomu, že zprávy vkládané do front systému IBM MQ nejsou zpřístupněny jiným aplikacím systému IBM MQ , dokud nejsou potvrzeny, musí být buď umístěny mimo synchronizační bod, nebo musí být aplikace IMS rozdělena do dvou transakcí.

Pokud dotaz zahrnuje vložení jediné zprávy, můžete použít volbu *no syncpoint* ; pokud je však dotaz složitější nebo pokud jsou zahrnuty aktualizace prostředků, může dojít k problémům s konzistencí, pokud dojde k selhání a nepoužijete funkci syncpoint.

Chcete-li to překonat, můžete rozdělit transakce produktu IMS MPP pomocí volání MQI pomocí přepínače zpráv program-program; informace o tom viz *IMS ISC (Intersystem Communication)* . To umožňuje implementaci dotazovacího programu 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
```

z/OS *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 IMS.

Pomocí následujících odkazů získáte další informace o psaní aplikací mostu IMS v systému IBM MQ for z/OS:

- “Jak most IMS pracuje se zprávami” na stránce 71
- “Zápis IMS transakčních programů prostřednictvím IBM MQ” na stránce 880

Související pojmy

“Psaní aplikací IMS pomocí IBM MQ” na stránce 66
Při používání produktu IBM MQ v aplikacích IMS je třeba vzít v úvahu další aspekty. Patří mezi ně volání rozhraní API produktu MQ a mechanizmus používaný pro synchronizační bod.

z/OS *Jak most IMS pracuje se zprávami*

Používáte-li most IBM MQ - IMS k odesílání zpráv do aplikace IMS , je třeba sestavit zprávy ve speciálním formátu.

Zprávy musíte také vložit do front systému IBM MQ , které byly definovány s paměťovou třídou, která určuje skupinu XCF a název člena cílového systému IMS . Tyto fronty se nazývají MQ-IMS nebo jednoduše **fronty mostu** .

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

Uživatel se nemusí přihlásit k produktu IMS před odesláním zpráv do aplikace IMS . ID uživatele v poli *UserIdentifier* struktury MQMD se používá pro kontrolu zabezpečení. Úroveň kontroly je určena při připojení produktu IBM MQ k serveru IMSa je popsána v tématu Řízení přístupu k aplikaci pro most IMS. To umožňuje, aby bylo implementováno pseudopřihlášení.

Most IBM MQ - IMS přijímá následující typy zpráv:

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

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

Poznámka:

1. Hranaté závorky [] představují volitelné vícenásobné segmenty.
2. Nastavte pole *Format* struktury MQMD na MQFMT_IMS pro použití struktury MQIIH.

- Zprávy obsahující data transakce IMS , ale bez struktury MQIIH:

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

Produkt IBM MQ ověří data zprávy a ujistí se, že součet bajtů LL plus délka MQIIH (je-li přítomna) se rovná délce zprávy.

Když most IBM MQ - IMS získá zprávy z front mostu, zpracuje je takto:

- Pokud zpráva obsahuje strukturu MQIIH, most ověří MQIIH (viz MQIIH), sestaví záhlaví OTMA a odešle zprávu do IMS. Kód transakce je uveden ve vstupní zprávě. Pokud se jedná o LTERM, produkt IMS odpoví zprávou DFS1288E . Pokud kód transakce představuje příkaz, IMS provede příkaz; jinak se zpráva zařadí do fronty IMS pro transakci.
- Pokud zpráva obsahuje transakční data IMS , ale nemá strukturu MQIIH, most IMS vytvoří následující předpoklady:
  - Kód transakce je v bajtech 5 až 12 uživatelských dat
  - Transakce je v nekonverzačním režimu
  - Transakce je v režimu potvrzení 0 (commit-then-send)
  - Parametr *Format* v deskriptoru MQMD se používá jako parametr *MFSMapName* (na vstupu).
  - Režim zabezpečení je MQISS_CHECK

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

Most IBM MQ - IMS používá pro každou frontu IBM MQ jednu nebo dvě propojení procesů:

- Synchronizovaná propojení procesů se používá pro všechny zprávy používající režim potvrzení 0 (COMMIT_THEN_SEND) (tyto zprávy se zobrazují s hodnotou SYN v poli stavu příkazu IMS /DIS TMEMBER client TPIPE xxxx).
- Nesynchronizovaná Tpipe se používá pro všechny zprávy používající režim potvrzení 1 (SEND_THEN_COMMIT)

Propojení procesů jsou vytvořena produktem IBM MQ při prvním použití. Nesynchronizované propojení procesů existuje, dokud není produkt IMS restartován. Synchronizovaná propojení procesů existují až do studeného spuštění produktu IMS . Tyto propojení procesů nemůžete odstranit sami.

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

- “Mapování zpráv IBM MQ na typy transakcí IMS” na stránce 72
- “Pokud zprávu nelze vložit do fronty IMS” na stránce 72
- “Kódy zpětné vazby mostu IMS” na stránce 73
- “Pole MQMD ve zprávách z mostu IMS” na stránce 73
- “Pole MQIIH ve zprávách z mostu IMS” na stránce 74
- “Odpovědět na zprávy od IMS” na stránce 75
- “Použití alternativních PCB odpovědi v transakcích IMS” na stránce 75
- “Odesílání nevyžádaných zpráv z IMS” na stránce 76
- “Segmentace zpráv” na stránce 76
- “Převod dat pro zprávy do a z mostu IMS” na stránce 76

Související pojmy

“Zápis IMS transakčních programů prostřednictvím IBM MQ” na stránce 880

Kódování požadované pro zpracování IMS transakcí prostřednictvím IBM MQ závisí na formátu zprávy požadovaném transakcí IMS a rozsahu odpovědí, které může vrátit. Existuje však několik bodů, které je třeba zvážit, když vaše aplikace zpracovává informace o formátování obrazovky IMS .

z/OS *Mapování zpráv IBM MQ na typy transakcí IMS*

Tabulka popisující mapování zpráv IBM MQ na typy transakcí IMS .

Tabulka 132. Způsob mapování zpráv produktu IBM MQ na typy transakcí IMS

IBM MQ typ zprávy	**Commit-then-send (režim 0)- používá synchronizované IMS Tpipes**	**Send-then-commit (režim 1)- používá nesynchronizované IMS Tpipes**
Trvalé zprávy IBM MQ	• Zotavitelné transakce s plnými funkcemi • Nezotavitelné transakce jsou odmítnuty IMS	• Rychlé transakce • Konverzační transakce • Transakce s plnou funkcí
Přechodné zprávy IBM MQ	• Nezotavitelné transakce s plnými funkcemi • Obnovitelné transakce jsou povoleny s produktem IMS V8 a opravou APAR PQ61404 a všemi novějšími verzemi produktu IMS	• Rychlé transakce • Konverzační transakce • Transakce s plnou funkcí

Poznámka: Příkazy IMS nemohou používat trvalé zprávy IBM MQ s režimem potvrzení 0. Další informace viz Režim potvrzení (commitMode) .

z/OS *Pokud zprávu nelze vložit do fronty IMS*

Informace o akcích, které je třeba provést v případě, že zprávu nelze vložit do fronty IMS .

Pokud zprávu nelze vložit do fronty IMS , IBM MQprovede následující akci:

- Pokud zprávu nelze vložit do souboru IMS , protože je neplatná, je vložena do fronty nedoručených zpráv a zpráva je odeslána do systémové konzoly.
- Pokud je zpráva platná, ale je odmítnuta produktem IMS, produkt IBM MQ odešle chybovou zprávu na systémovou konzolu, zpráva obsahuje chybový kód IMS a zpráva IBM MQ je vložena do fronty nedoručených zpráv. Je-li IMS chybový kód 001A, IMS odešle zprávu IBM MQ obsahující příčinu selhání do fronty pro odpověď.

Poznámka: Za výše uvedených okolností platí, že pokud produkt IBM MQ nemůže z jakéhokoli důvodu vložit zprávu do fronty nedoručených zpráv, vrátí se zpráva do původní fronty IBM MQ . Do systémové konzoly se odešle chybová zpráva a z této fronty se neodešlou žádné další zprávy.

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

- Zastavte a restartujte propojení procesů v adresáři IMS odpovídající frontě.
- Změnit frontu na GET (DISABLED) a znovu na GET (ENABLED)
- Zastavte a restartujte IMS nebo OTMA
- Zastavte a restartujte subsystém IBM MQ .

- Pokud je zpráva odmítnuta produktem IMS kvůli jiné chybě než kvůli chybě zprávy, vrátí se zpráva IBM MQ do původní fronty, produkt IBM MQ zastaví zpracování fronty a na systémovou konzolu se odešle chybová zpráva.

  Je-li vyžadována zpráva sestavy výjimek, most ji vloží do fronty pro odpověď s oprávněním původce. Pokud zprávu nelze vložit do fronty, zpráva sestavy se vloží do fronty nedoručených zpráv s oprávněním mostu. Pokud jej nelze vložit do fronty DLQ, bude vyřazen.

z/OS *Kódy zpětné vazby mostu IMS*

IMS chybové kódy jsou obvykle vypisovány v hexadecimálním formátu ve zprávách konzoly IBM MQ , jako např. CSQ2001I (například chybový kód 0x001F). Kódy zpětné vazby IBM MQ uvedené v záhlaví nedoručených zpráv vložených do fronty nedoručených zpráv jsou desetinná čísla.

Kódy zpětné vazby mostu IMS jsou v rozsahu 301 až 399 nebo 600 až 855 pro chybový kód NACK 0x001A. Jsou mapovány ze smyslových kódů IMS-OTMA takto:

1. Kód chyby IMS-OTMA je převeden z hexadecimálního čísla na desetinné číslo.
2. 300 se přidá k číslu, které je výsledkem výpočtu v 1, s uvedením kódu IBM MQ *Feedback* .
3. Chybový kód IMS-OTMA 0x001A, desetinný 26 je speciální případ. Vygeneruje se kód *Zpětná vazba* v rozsahu 600-855.
   a. Kód příčiny IMS-OTMA je převeden z hexadecimálního čísla na desetinné číslo.
   b. 600 se přidá k číslu, které je výsledkem výpočtu v a, s kódem IBM MQ *Zpětná vazba* .

Informace o chybových kódech IMS-OTMA viz chybové kódy OTMA pro zprávy NAK.

z/OS *Pole MQMD ve zprávách z mostu IMS*

Informace o polích MQMD ve zprávách z mostu IMS .

Modul MQMD původní zprávy je přenášen produktem IMS v sekci Uživatelská data záhlaví OTMA. Pokud zpráva pochází z produktu IMS, je sestavena uživatelskou procedurou pro rozlišení cíle systému IMS . Modul MQMD zprávy přijaté z produktu IMS je sestaven následujícím způsobem:

StrucID
"MD"

Verze
MQMD_VERSION_1

Sestava
MQRO_NONE

MsgType
MQMT_REPLY

Vypršení
Pokud je v poli Příznaky MQIIH nastavena hodnota MQIH_PASS_EXPIRATION, bude toto pole obsahovat zbývající čas vypršení platnosti, jinak bude nastaveno 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, pokud je MQMD.Format vstupní zprávy je MQFMT_IMS, jinak IOPCB.MODNAME

Priorita
MQMD.Priority vstupní zprávy

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

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

CorrelId
MQMD.CorrelId ze vstupní zprávy, pokud MQRO_PASS_CORREL_ID, jinak MQMD.MsgId ze vstupní zprávy (výchozí)

BackoutCount
0

ReplyToQ
Mezery

ReplyToQMgr
Mezery (správce front během operace MQPUT nastavil na lokální název správce front)

UserIdentifier
MQMD.UserIdentifier vstupní zprávy

AccountingToken
MQMD.AccountingToken vstupní zprávy

ApplIdentityData
MQMD.ApplIdentityData vstupní zprávy

PutApplType
MQAT_XCF-pokud nedošlo k chybě, jinak MQAT_BRIDGE

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

PutDate
Datum, kdy byla zpráva vložena

PutTime
Čas, kdy byla zpráva vložena

ApplOriginData
Mezery

z/OS *Pole MQIIH ve zprávách z mostu IMS*

Získejte informace o polích MQIIH ve zprávách z mostu IMS .

MQIIH zprávy přijaté od IMS je sestaveno takto:

StrucId
"IIH"

Verze
1

StrucLength
84

Kódování
MQENC_NATIVE

CodedCharSetId
MQCCSI_Q_MGR

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

Příznaky
0

LTermOverride
Název LTERM (Tpipe) ze záhlaví OTMA

MFSMapName
Název mapy ze záhlaví OTMA

ReplyToFormát
Mezery

Ověřovatel
MQIIH.Authenticator vstupní zprávy, pokud je zpráva odpovědi vkládána do fronty mostu MQ-IMS , jinak prázdná hodnota.

ID TranInstance
ID konverzace/Token serveru ze záhlaví OTMA, pokud je v konverzaci. Ve verzích systému IMS starších než V14je toto pole vždy prázdné, pokud není v konverzaci. Od verze IMS V14 dále může toto pole nastavit IMS , i když není v konverzaci.

TranState
"C", pokud je v konverzaci, jinak prázdné

CommitMode
Režim potvrzení ze záhlaví OTMA ("0" nebo "1")

SecurityScope
Prázdný

Vyhrazeno
Prázdný

z/OS *Odpovědět na zprávy od IMS*

Když transakce IMS ISRTs na svůj IOPCB, zpráva je směrována zpět na původní LTERM nebo TPIPE.

Ty jsou v produktu IBM MQ vnímány jako zprávy odpovědi. Zprávy odpovědí z produktu IMS jsou vloženy do fronty pro odpověď určené v původní zprávě. Pokud zprávu nelze vložit do fronty pro odpověď, je vložena do fronty nedoručených zpráv s použitím oprávnění mostu. Pokud zprávu nelze vložit do fronty nedoručených zpráv, odešle se na adresu IMS negativní potvrzení, že zprávu nelze přijmout. Odpovědnost za zprávu se pak vrátí na IMS. Používáte-li režim potvrzení 0, nebudou zprávy z tohoto propojení procesů odesílány do mostu a zůstanou ve frontě IMS . To znamená, že do restartování nebudou odesílány žádné další zprávy. Pokud používáte režim potvrzení 1, může pokračovat jiná práce.

Pokud má odpověď strukturu MQIIH, její typ formátu je MQFMT_IMS; pokud ne, její typ formátu je určen názvem IMS MOD použitým při vkládání zprávy.

z/OS *Použití alternativních PCB odpovědi v transakcích IMS*

Když transakce IMS používá alternativní PCB odezvy (ISRTs pro ALTPCB, nebo vydá volání CHNG pro modifikovatelný PCB), je vyvolána uživatelská procedura před směrováním (DFSYPRX0), aby se zjistilo, zda má být zpráva přesměrována.

Pokud má být zpráva přesměrována, je vyvolána uživatelská procedura rozpoznání místa určení (DFSYDRU0), která potvrdí místo určení a připraví informace v záhlaví. Informace o těchto uživatelských programech naleznete v tématu Použití uživatelských procedur OTMA v produktu IMS a Uživatelský program před směrováním DFSYPRX0 .

Není-li v uživatelských procedur provedena akce, všechny výstupy z transakcí IMS zahájených ze správce front IBM MQ , ať už do IOPCB nebo ALTPCB, budou vráceny do stejného správce front.

z/OS *Odesílání nevyžádaných zpráv z IMS*

Chcete-li odeslat zprávy z IMS do fronty IBM MQ , musíte vyvolat transakci IMS , kterou ISRTs odešle do ALTPCB.

Pro směrování nevyžádaných zpráv z produktu IMS a sestavení uživatelských dat OTMA je třeba napsat uživatelské procedury pro předběžné směrování a rozlišení cíle, aby bylo možné správně sestavit MQMD zprávy. Informace o těchto uživatelských programech viz Uživatelská procedura před směrováním DFSYPRX0 a Uživatelská procedura pro rozlišení cíle .

Poznámka: Most IBM MQ - IMS neví, zda je zpráva, kterou obdrží, odpovědí nebo nevyžádanou zprávou. Zpracovává zprávu stejným způsobem v každém případě, sestaví MQMD a MQIIH odpovědi na základě UserData OTMA, která přišla se zprávou.

Nevyžádané zprávy mohou vytvářet nové Troury. Pokud například existující transakce IMS přepnula na nový LTERM (například PRINT01), ale implementace vyžaduje, aby byl výstup doručen prostřednictvím OTMA, vytvoří se nový Tpipe (v tomto příkladu s názvem PRINT01). Standardně se jedná o nesynchronizované propojení procesů. Pokud implementace vyžaduje, aby byla zpráva obnovitelná, nastavte příznak výstupu ukončení rozpoznání cíle. Další informace viz příručka *IMS Customization Guide* .

z/OS *Segmentace zpráv*

Transakce IMS můžete definovat jako očekávaný vstup s jedním nebo více segmenty.

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

Maximální délka datového segmentu LLZZ je definována pomocí IMS/OTMA (32767 bajtů). Celková délka zprávy IBM MQ je součtem bajtů LL plus délka struktury MQIIH.

Všechny segmenty odpovědi jsou obsaženy v jediné zprávě IBM MQ .

Existuje další omezení omezení 32 kB pro zprávy s formátem MQFMT_IMS_VAR_STRING. Když jsou data ve zprávě ASCII-mixed CCSID převedena na zprávu EBCDIC-mixed CCSID, je shift-in byte nebo shift-out byte přidán pokaždé, když dojde k přechodu mezi znaky SBCS a DBCS. Omezení 32 kB platí pro maximální velikost zprávy. To znamená, že protože pole LL ve zprávě nesmí překročit 32 kB, nesmí zpráva překročit 32 kB včetně všech znaků shift-in a shift-out. Aplikace, která zprávu sestavuje, to musí povolit.

z/OS *Převod dat pro zprávy do a z mostu IMS*

Převod dat provádí buď distribuovaný prostředek front (který může volat nezbytné uživatelské procedury), nebo agent front v rámci skupiny (který nepodporuje použití uživatelských procedur), když vkládá zprávu do cílové fronty, která má pro svou paměťovou třídu definovány informace XCF. K převodu dat nedojde, když je zpráva doručena do fronty prostřednictvím publikování/odběru.

Všechny potřebné uživatelské procedury musí být k dispozici pro službu distribuovaného řazení do 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 .

Dojde-li k chybám při převodu, zpráva se vloží do fronty nepřevedené; to má za následek, že most IBM MQ - IMS bude nakonec považován za chybu, protože most nerozpozná formát záhlaví. Dojde-li k chybě převodu, odešle se chybová zpráva na konzolu z/OS .

Podrobné informace o převodu dat obecně viz “Zápis uživatelských procedur převodu dat” na stránce 950 .

Odesílání zpráv do mostu IBM MQ - IMS

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

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

Pokud jsou vaše data (jiná než LLZZs) všechna znaková data (MQCHAR), použijte jako název formátu (v MQIIH nebo MQMD, podle potřeby) vestavěný formát MQFMT_IMS_VAR_STRING. Jinak použijte vlastní název formátu, v takovém případě musíte také poskytnout uživatelskou proceduru pro převod dat pro váš formát. Uživatelská procedura musí zpracovat převod LLZZs ve vaší zprávě, kromě dat samotných (ale nemusí zpracovat žádnou MQIIH na začátku zprávy).

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

Příjem zpráv z mostu IBM MQ - IMS

Je-li v původní zprávě, kterou odesíláte do produktu IMS, uvedena struktura MQIIH, je uvedena i ve zprávě odpovědi.

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

- Máte-li strukturu MQIIH pro původní zprávu, zadejte požadovaný formát pro zprávu odpovědi do pole MQIIH *ReplytoFormat* původní zprávy. Tato hodnota je umístěna do pole MQIIH *Format* zprávy odpovědi. To je užitečné zejména v případě, že všechna výstupní data jsou ve formátu LLZZ < znaková data >.
- Pokud nemáte strukturu MQIIH ve své původní zprávě, uveďte formát, který chcete pro zprávu odpovědi, jako název MFS MOD v ISRT aplikace IMS pro IOPCB.

z/OS *Zápis IMS transakčních programů prostřednictvím IBM MQ*

Kódování požadované pro zpracování IMS transakcí prostřednictvím IBM MQ závisí na formátu zprávy požadovaném transakcí IMS a rozsahu odpovědí, které může vrátit. Existuje však několik bodů, které je třeba zvážit, když vaše aplikace zpracovává informace o formátování obrazovky IMS .

Když je transakce IMS spuštěna z obrazovky 3270, zpráva prochází produktem IMS Message Format Services. To může odebrat veškerou závislost terminálu z datového proudu, který transakce vidí. Když je transakce spuštěna prostřednictvím OTMA, MFS se neúčastní. Pokud je logika aplikace implementována v MFS, musí být znovu vytvořena v nové aplikaci.

V některých transakcích systému IMS může aplikace koncového uživatele upravit určité chování obrazovky 3270, například zvýrazněním pole, pro které byla zadána neplatná data. Tento typ informací je komunikován přidáním dvoubajtového pole atributu do zprávy IMS pro každé pole obrazovky, které musí program upravit.

Pokud tedy kódujete aplikaci tak, aby napodobovala 3270, musíte vzít v úvahu tato pole při vytváření nebo přijímání zpráv.

Možná budete muset kódovat informace ve svém programu ke zpracování:

- Která klávesa je stisknuta (například Enter a PF1)
- Kde je kurzor, když je zpráva předána vaší aplikaci
- Zda byla pole atributů nastavena aplikací IMS
  - Vysoká, normální nebo nulová intenzita
  - Barva
  - Zda IMS očekává pole zpět při příštím stisknutí klávesy Enter
- Zda aplikace IMS použila v polích znaky null (X'3F').

Pokud zpráva systému IMS obsahuje pouze znaková data (kromě segmentu LLZZ-data) 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 segmentu LLZZ-data) a **nepoužíváte** strukturu MQIIH, nastavte formát MQMD na MQFMT_IMS_VAR_STRING a ujistěte se, že vaše aplikace IMS při odpovídání uvádí MODname MQFMT_IMS_VAR_STRING. Pokud se vyskytne problém (například uživatel, který nemá oprávnění k použití transakce) a produkt IMS odešle chybovou zprávu, bude mít název MODname ve tvaru DFSMOx, kde x je číslo v rozsahu 1 až 5. Toto je vloženo do MQMD.Format.

Pokud zpráva systému IMS obsahuje binární, sbalená nebo data s pohyblivou řádovou čárkou (kromě datového segmentu LLZZ), kódujte vlastní rutiny pro převod dat. Informace o formátování obrazovky IMS naleznete v tématu *IMS/ESA Programování aplikací: správce transakcí* .

Při zápisu kódu pro zpracování IMS transakcí prostřednictvím IBM MQzvažte následující témata.

- “Zápis IBM MQ aplikací pro vyvolání IMS konverzačních transakcí” na stránce 881
- “Psaní programů obsahujících příkazy IMS” na stránce 881
- “Spouątění” na stránce 881

Zápis IBM MQ aplikací pro vyvolání IMS konverzačních transakcí

Tyto informace použijte jako vodítko pro aspekty při psaní aplikace IBM MQ pro vyvolání IMS konverzačních transakcí.

Při psaní aplikace, která vyvolává konverzaci produktu IMS , zvažte následující skutečnosti:

- Do zprávy aplikace zahrňte strukturu MQIIH.
- Nastavte parametr *CommitMode* v MQIIH na hodnotu MQICM_SEND_THEN_COMMIT.
- Chcete-li vyvolat novou konverzaci, nastavte parametr *TranState* v MQIIH na hodnotu MQITS_NOT_IN_CONVERSATION.
- Chcete-li vyvolat druhý a následující postup konverzace, nastavte parametr *TranState* na hodnotu MQITS_IN_CONVERSATION a parametr *TranInstanceId* nastavte na hodnotu pole vrácenou v předchozím kroku konverzace.
- V produktu IMS není snadný způsob, jak najít hodnotu *TranInstanceId*, pokud ztratíte původní zprávu odeslanou z produktu IMS.
- Aplikace musí zkontrolovat *TranState* zpráv z IMS , aby zkontrolovala, zda transakce IMS ukončila konverzaci.
- K ukončení konverzace můžete použít /EXIT. Musíte také citovat *TranInstanceId*, nastavit *TranState* na MQITS_IN_CONVERSATION a použít frontu IBM MQ , ve které se konverzace provádí.
- Nemůžete použít /HOLD nebo /REL k zadržení nebo uvolnění konverzace.
- Konverzace vyvolané prostřednictvím mostu IBM MQ - IMS jsou ukončeny, pokud je IMS restartován.

Psaní programů obsahujících příkazy IMS

Aplikační program může sestavit IBM MQ zprávu ve tvaru LLZZ*příkaz*místo transakce, kde *příkaz* je ve tvaru /DIS TRAN PART nebo /DIS POOL ALL.

Většina příkazů IMS může být zadána tímto způsobem; podrobnosti viz *IMS V11 Komunikace a připojení* . Výstup příkazu je přijat ve zprávě odpovědi IBM MQ v textové podobě, jako by byl odeslán na terminál 3270 pro zobrazení.

OTMA implementovala speciální formu příkazu IMS display transaction, který vrací vytvořenou 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 hodnotu MQITS_ARCHITECTED. Produkt IMS zpracuje příkaz a vrátí odpověď v architektonické podobě. Odpověď s architekturou obsahuje všechny informace, které lze nalézt v textové podobě výstupu, a jednu další informaci: zda je transakce definována jako obnovitelná nebo neobnovitelná.

Spouątění

Most IBM MQ - IMS nepodporuje zprávy spouštěče.

Definujete-li inicializační frontu, která používá paměťovou třídu s parametry XCF, budou zprávy vkládané do této fronty při přechodu na most odmítnuty.

Psaní procedurálních aplikací klienta

Co potřebujete vědět, abyste mohli psát klientské aplikace na systému IBM MQ pomocí procedurálního jazyka.

Aplikace lze sestavit a spustit v prostředí klienta IBM MQ . Aplikace musí být sestavena a propojena s použitými IBM MQ MQI client . Způsob, jakým jsou aplikace sestavovány a propojeny, se liší podle použité platformy a programovacího jazyka. Informace o tom, jak sestavovat klientské aplikace, viz “Sestavení aplikací pro IBM MQ MQI clients” na stránce 887.

Aplikaci IBM MQ můžete spustit jak v úplném prostředí IBM MQ , tak v prostředí IBM MQ MQI client , aniž byste změnili kód, za předpokladu, že jsou splněny určité podmínky. Další informace o spouštění aplikací v prostředí klienta IBM MQ naleznete v části “Spuštění aplikací v prostředí IBM MQ MQI client” na stránce 889.

Používáte-li rozhraní fronty zpráv (MQI) k zápisu aplikací pro spouštění v prostředí IBM MQ MQI client , je třeba během volání MQI zavést další ovládací prvky, které zajistí, že zpracování aplikace IBM MQ nebude přerušeno. Další informace o těchto ovládacích prvcích viz “Použití rozhraní MQI v aplikaci klienta” na stránce 883.

Informace o přípravě a spouštění dalších typů aplikací jako klientských aplikací naleznete v následujících tématech:

- “Příprava a spuštění aplikací CICS a Tuxedo” na stránce 901
- “Příprava a spuštění aplikací serveru Microsoft Transaction Server” na stránce 47
- “Příprava a spuštění aplikací IBM MQ JMS” na stránce 903

Související pojmy

“Koncepty vývoje aplikací” na stránce 7
K psaní aplikací IBM MQ můžete použít výběr procedurálních nebo objektově orientovaných jazyků. Než začnete navrhovat a psát aplikace IBM MQ , seznamte se se základními koncepty produktu IBM MQ .

“Vyvíjení aplikací pro IBM MQ” na stránce 5
Můžete vyvíjet aplikace pro odesílání a příjem zpráv a pro správu správců front a souvisejících prostředků. Produkt IBM MQ podporuje aplikace napsané v mnoha různých jazycích a rámcích.

“Aspekty návrhu pro aplikace IBM MQ” na stránce 47
Když jste se rozhodli, jak mohou vaše aplikace využívat platformy a prostředí, která máte k dispozici, musíte se rozhodnout, jak používat funkce nabízené produktem IBM MQ.

“Psaní procedurální žádosti o zařazení do fronty” na stránce 699
Prostřednictvím těchto informací můžete získat informace o psaní aplikací řazení do front, o připojování a odpojování od správce front, o publikování/odběru a o otevírání a zavírání objektů.

“Zápis aplikací publikování/odběru” na stránce 782
Začněte psát aplikace publikování/odběru IBM MQ .

“Sestavení procedurální aplikace” na stránce 967
Aplikaci IBM MQ můžete napsat v jednom z několika procedurálních jazyků a spustit ji na několika různých platformách.

“Zpracování chyb procedurálních programů” na stránce 1004
Tyto informace vysvětlují chyby související s voláními rozhraní MQI aplikací, a to buď při volání, nebo při doručení zprávy do konečného místa určení.

Související úlohy

“Použití ukázkových procedurálních programů IBM MQ” na stránce 1023
Tyto ukázkové programy jsou napsány v procedurálních jazycích a ukazují typické použití rozhraní MQI (Message Queue Interface). Programy IBM MQ na různých platformách.

Použití rozhraní MQI v aplikaci klienta

Tato kolekce témat zohledňuje rozdíly mezi zápisem aplikace IBM MQ , která má být spuštěna v klientském prostředí rozhraní fronty zpráv (MQI), a spuštěním v úplném prostředí správce front produktu IBM MQ .

Při návrhu aplikace zvažte, jaké ovládací prvky je třeba zavést během volání MQI, abyste se ujistili, že zpracování aplikace IBM MQ nebude narušeno.

Než budete moci spustit aplikace, které používají rozhraní MQI, musíte vytvořit určité objekty IBM MQ . Další informace naleznete v tématu Aplikační programy používající rozhraní MQI.

Omezení velikosti zprávy v klientské aplikaci

Správce front má maximální délku zprávy, ale maximální velikost zprávy, kterou můžete přenést z klientské aplikace, je omezena definicí kanálu.

Atribut maximální délky zprávy (MaxMsgLength) správce front je maximální délkou zprávy, kterou může daný správce front zpracovat.

Multi V systému Multiplatformsmůžete zvýšit atribut maximální délky zprávy správce front. Další informace viz ALTER QMGR.

Pomocí volání MQINQ můžete zjistit hodnotu délky MaxMsgpro správce front.

Pokud se změní atribut MaxMsgLength, neprovede se žádná kontrola, zda již nejsou fronty, a dokonce ani zprávy, jejichž délka je větší než nová hodnota. Po změně tohoto atributu restartujte aplikace a kanály, abyste se ujistili, že se změna projevila. Poté není možné generovat žádné nové zprávy, které by překročily délku MaxMsgsprávce 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 tato, vrátí se aplikaci kód chyby. Parametr maximální velikosti zprávy definice kanálu nemá vliv na maximální velikost zprávy, kterou lze spotřebovat pomocí MQCB přes připojení klienta.

Související pojmy

“Použití MQCONNX” na stránce 887
Pomocí volání MQCONNX můžete určit strukturu definice kanálu (MQCD) ve struktuře MQCNO.

Související odkazy

Maximální délka zprávy (MAXMSGL)

POZMĚNIT KANÁL

2010 (07DA) (RC2010): MQRC_DATA_LENGTH_ERROR

Výběr CCSID klienta nebo serveru

Použijte lokální identifikátor kódované znakové sady (CCSID) pro klienta. Správce front provádí nezbytný převod. K přepsání CCSID můžete použít proměnnou prostředí **MQCCSID** . Pokud vaše aplikace provádí více PUTů, mohou být pole CCSID a kódování MQMD po dokončení prvního PUT přepsána.

Data předaná přes rozhraní fronty zpráv (MQI) z aplikace do stubu klienta musí být v lokálním CCSID, zakódovaném pro IBM MQ MQI client. Pokud připojený správce front vyžaduje převod dat, provede převod kód podpory klienta ve správci front.

V produktu IBM WebSphere MQ 7.0 a novějších verzích může klient produktu Java provést převod, pokud to správce front nemůže provést. Viz téma “IBM MQ classes for Java připojení klientů” na stránce 360.

Kód klienta předpokládá, že znaková data překračující rozhraní MQI v klientu jsou v CCSID konfigurovaném pro danou pracovní stanici. Pokud je tento CCSID nepodporovaný CCSID nebo není požadovaný CCSID, může být přepsán pomocí proměnné prostředí **MQCCSID** pomocí jednoho z těchto příkazů:

- Windows

```
SET MQCCSID=850
```

- Linux AIX

```
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: Předpoklad o kódové stránce 850 se nevztahuje na data aplikace ve zprávě.

Pokud vaše aplikace provádí více sad PUTs, které zahrnují záhlaví IBM MQ za deskriptorem zprávy (MQMD), mějte na paměti, že pole CCSID a kódování deskriptoru MQMD jsou po dokončení první operace PUT přepsána.

Po prvním příkazu PUT tato pole obsahují hodnotu používanou připojeným správcem front pro převod záhlaví IBM MQ . Ujistěte se, že vaše aplikace resetuje hodnoty na hodnoty, které vyžaduje.

Použití MQINQ v aplikaci klienta

Některé hodnoty dotazované pomocí MQINQ jsou upraveny kódem klienta.

CCSID
: je nastaven na CCSID klienta, nikoli na CCSID správce front.

MaxMsgLength
: je snížena, pokud je omezena definicí kanálu. To bude nižší z:

  - Hodnota definovaná v definici fronty nebo
  - Hodnota definovaná v definici kanálu

Další informace viz MQINQ.

Použití koordinace synchronizačních bodů v klientské aplikaci

Aplikace spuštěná v základním klientu může vydávat příkazy MQCMIT a MQBACK, ale rozsah řízení synchronizačního bodu je omezen na prostředky MQI. Můžete použít externího správce transakcí s rozšířeným transakčním klientem.

V rámci produktu IBM MQje jednou z rolí správce front řízení synchronizačních bodů v rámci aplikace. Pokud je aplikace spuštěna na základním klientovi IBM MQ , může vydat příkazy MQCMIT a MQBACK, ale rozsah řízení synchronizačního bodu je omezen na prostředky MQI. Příkaz IBM MQ MQBEGIN není platný v prostředí základního klienta.

Aplikace spuštěné v úplném prostředí správce front na serveru mohou koordinovat více prostředků (například databáze) prostřednictvím monitoru transakcí. Na serveru můžete použít monitor transakcí dodávaný s produkty IBM MQ nebo jiný monitor transakcí, například CICS. Monitor transakcí nelze použít se základní klientskou aplikací.

Můžete použít externího správce transakcí s rozšířeným transakčním klientem IBM MQ . Viz Co je to rozšířený transakční klient? pro podrobnosti.

Použití dopředného čtení v aplikaci klienta

Můžete použít dopředné čtení na klientovi, chcete-li povolit dočasné zprávy, které mají být odeslány 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 zlepšit výkon klienta, který spotřebovává přechodné zprávy, a vyhnout se tak odesílání těchto zpráv požadavků, lze klienta nakonfigurovat tak, aby používal

dopředné čtení. Dopředné čtení umožňuje, aby byly zprávy odesílány klientovi, aniž by je aplikace musela požadovat.

Použití dopředného čtení může zlepšit výkon při příjmu dočasných zpráv z klientské aplikace. Toto zlepšení výkonu je k dispozici pro aplikace MQI i JMS . Klientské aplikace používající příkaz MQGET nebo asynchronní spotřebu těží ze zlepšení výkonu při spotřebě dočasných 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:

- 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 dopředné čtení, jsou zprávy odesílány do vyrovnávací paměti v klientovi, která se nazývá vyrovnávací paměť dopředného čtení. Klient má vyrovnávací paměť pro dopředné čtení pro každou frontu, kterou má otevřenou s povoleným dopředným čtením. Zprávy ve vyrovnávací paměti dopředného čtení nejsou trvalé. Klient pravidelně aktualizuje server s informacemi o množství dat, které spotřeboval.

Ne všechny návrhy klientských aplikací jsou vhodné pro použití dopředného čtení, protože ne všechny volby jsou podporovány pro použití. Některé volby musí být konzistentní mezi voláními MQGET, když je povoleno dopředné čtení. Pokud klient změní kritéria výběru mezi voláními MQGET, zprávy uložené ve vyrovnávací paměti dopředného čtení zůstanou ve vyrovnávací paměti dopředného čtení klienta. Další informace viz “Zlepšení výkonu dočasných zpráv” na stránce 761

Konfigurace dopředného čtení je řízena třemi atributy, MaximumSize, PurgeTimea UpdatePercentage, které jsou určeny v sekci MessageBuffer konfiguračního souboru klienta IBM MQ .

Použití asynchronního vložení v klientské aplikaci

Pomocí asynchronního vložení může aplikace vložit zprávu do fronty bez čekání na odpověď od správce front. Tuto funkci 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 potvrzení správce front, že zpracoval požadavek MQI. Můžete zlepšit výkon systému zpráv, zejména pro aplikace, které používají vazby klienta, a aplikace, které do fronty vkládají velký počet malých zpráv, a to tak, že místo toho budou zprávy vkládat asynchronně. Když aplikace vloží zprávu asynchronně, správce front nevrátí úspěch nebo selhání každého volání, ale místo toho můžete pravidelně kontrolovat chyby.

Chcete-li vložit zprávu do fronty asynchronně, použijte volbu MQPMO_ASYNC_RESPONSE v poli *Options* struktury MQPMO.

Pokud zpráva není vhodná pro asynchronní vložení, je vložena do fronty synchronně.

Při vyžádání asynchronní odezvy vložení pro MQPUT nebo MQPUT1nemusí CompCode a příčina MQCC_OK a MQRC_NONE nutně znamenat, že zpráva byla úspěšně vložena do fronty. Ačkoli úspěch nebo selhání jednotlivých volání MQPUT nebo MQPUT1 nemusí být vráceno okamžitě, první chybu, ke které došlo v rámci asynchronního volání, lze určit později prostřednictvím volání MQSTAT.

Další podrobnosti o parametru MQPMO_ASYNC_RESPONSE naleznete v tématu Volby MQPMO.

Ukázkový program Asynchronní vložení demonstruje některé z dostupných funkcí. Podrobnosti o funkcích a návrhu programu a o jeho spuštění viz “Ukázkový program Asynchronní vložení” na stránce 1041.

Použití sdílení konverzací v klientské aplikaci

V prostředí, kde je povoleno sdílení konverzací, mohou konverzace sdílet instanci kanálu MQI.

Sdílení konverzací je řízeno dvěma poli, obě s názvem SharingConversations, z nichž jedno je součástí struktury definice kanálu (MQCD) a jedno je součástí struktury parametru uživatelské procedury kanálu (MQCXP). Pole SharingConversations v MQCD je celočíselná hodnota určující maximální počet konverzací,

které mohou sdílet instanci kanálu přidruženou ke kanálu. Pole SharingConversations v MQCXP má logickou hodnotu, která určuje, zda je instance kanálu aktuálně sdílena.

V prostředí, kde není povoleno sdílení konverzací, nebudou nová připojení klienta určující identické tabulky MQCD sdílet instanci kanálu.

Nové připojení klientské aplikace bude sdílet instanci kanálu, pokud jsou splněny následující podmínky:

- Konce připojení klienta i připojení serveru instance kanálu jsou konfigurovány pro sdílení konverzací a tyto hodnoty nejsou přepsány ukončeními kanálu.
- Hodnota MQCD připojení klienta (zadaná ve volání MQCONNX klienta nebo z tabulky CCDT (Client Channel Definition Table)) přesně odpovídá hodnotě MQCD připojení klienta zadané ve volání MQCONNX klienta nebo z tabulky CCDT při prvním vytvoření existující instance kanálu. Všimněte si, že původní disk MQCD mohl být následně pozměněn ukončením nebo vyjednáním kanálu, ale byla provedena shoda s hodnotou, která byla dodána klientskému systému před provedením těchto změn.
- Limit konverzací sdílení na straně serveru není překročen.

Pokud nové připojení klientské aplikace splňuje kritéria pro spuštění sdílení instance kanálu s jinými konverzacemi, je toto rozhodnutí učiněno před voláním uživatelských procedur pro tuto konverzaci. Ukončení takové konverzace nemůže změnit skutečnost, že sdílí instanci kanálu s jinými konverzacemi. Pokud neexistují žádné instance kanálu odpovídající nové definici kanálu, je připojena nová instance kanálu.

Vyjednávání kanálu probíhá pouze pro první konverzaci na instanci kanálu. Vyjednané hodnoty pro instanci kanálu jsou v této fázi opraveny a nelze je změnit při spuštění následných konverzací. Ověření TLS se také vyskytuje pouze pro první konverzaci.

Pokud je hodnota MQCD SharingConversations změněna během inicializace jakéhokoli zabezpečení, odeslání nebo přijetí uživatelských procedur pro první konverzaci na soketu buď v připojení klienta, nebo na konci připojení serveru instance kanálu, použije se nová hodnota, kterou má po inicializaci všech těchto uživatelských procedur, k určení hodnoty sdílených konverzací pro instanci kanálu (nejnižší hodnota má přednost).

Je-li vyjednaná hodnota pro sdílení konverzací nulová, instance kanálu se nikdy nesdílí. Další ukončovací programy, které nastaví toto pole na nulu, budou podobně spuštěny na vlastní instanci kanálu.

Je-li vyjednaná hodnota pro sdílení konverzací větší než nula, MQCXP SharingConversations je pro následná volání uživatelských procedur nastavena na hodnotu TRUE, což znamená, že ostatní uživatelské programy v této instanci kanálu lze zadat současně s tímto.

Při psaní programu uživatelské procedury kanálu zvažte, zda bude spuštěn na instanci kanálu, která může zahrnovat sdílení konverzací. Pokud může instance kanálu zahrnovat sdílení konverzací, zvažte vliv změny polí MQCD na jiné instance uživatelské procedury kanálu; všechna pole MQCD mají společné hodnoty ve všech konverzacích sdílení. Pokud se po vytvoření instance kanálu uživatelské programy pokusí změnit pole MQCD, mohou narazit na problémy, protože jiné instance uživatelských programů spuštěných na instanci kanálu se mohou současně pokoušet o změnu stejných polí. Pokud by se tato situace mohla vyskytnout u vašich uživatelských programů, musíte serializovat přístup k MQCD ve vašem uživatelském kódu.

Pokud pracujete s kanálem, který je definován pro sdílení konverzací, ale nechcete, aby se sdílení vyskytlo v určité instanci kanálu, nastavte při inicializaci uživatelské procedury kanálu v první konverzaci v instanci kanálu hodnotu MQCD SharingConversations na hodnotu 1 nebo 0. Vysvětlení hodnot parametru SharingConversationsnaleznete v tématu SharingConversations .

Příklad

Sdílení konverzací je povoleno.

Používáte definici kanálu připojení klienta, která určuje uživatelský program.

Při prvním spuštění tohoto kanálu uživatelský program při inicializaci změní některé parametry MQCD. Tyto akce jsou ovlivněny kanálem, takže definice, se kterou je kanál spuštěn, se nyní liší od definice, která byla původně dodána. Parametr MQCXP SharingConversations je nastaven na hodnotu TRUE.

Při příštím připojení aplikace pomocí tohoto kanálu bude konverzace spuštěna na instanci kanálu, která byla spuštěna dříve, protože má stejnou původní definici kanálu. Instance kanálu, ke které se aplikace připojuje podruhé, je stejná jako při prvním připojení. V důsledku toho používá definice, které byly změněny uživatelským programem. Je-li uživatelský program inicializován pro druhou konverzaci, ačkoli může měnit pole MQCD, kanál s nimi nepracuje. Tyto stejné charakteristiky platí pro všechny následné konverzace, které sdílejí instanci kanálu.

Použití MQCONNX

Pomocí volání MQCONNX můžete určit strukturu definice kanálu (MQCD) ve struktuře MQCNO.

To umožňuje volající aplikaci klienta určit definici kanálu připojení klienta za běhu. Další informace naleznete v tématu Vytvoření kanálu připojení klienta na serveru IBM MQ MQI client pomocí MQCNO. Při použití MQCONNX závisí volání vydané na serveru na úrovni serveru a konfiguraci modulu listener.

Při použití MQCONNX z klienta jsou ignorovány následující volby:

- MQCNO_STANDARDNÍ_VAZBA
- MQCNO_FASTPATH_BINDING

Struktura MQCD, kterou můžete použít, závisí na čísle verze MQCD, kterou používáte. Informace o verzích MQCD (MQCD_VERSION) naleznete v tématu Verze MQCD. Strukturu MQCD můžete použít například k předání programů uživatelské procedury kanálu na server. Používáte-li MQCD verze 3 nebo novější, můžete tuto strukturu použít k předání pole uživatelských procedur serveru. Tuto funkci můžete použít k provedení více než jedné operace na stejné zprávě, jako je šifrování a komprese, přidáním uživatelské procedury pro každou operaci, spíše než úpravou existující uživatelské procedury. Pokud nezadáte pole ve struktuře MQCD, budou zkontrolována pole s jednou uživatelskou procedurou. Další informace o programech pro uživatelské procedury kanálu naleznete v části “Programy uživatelské procedury kanálu pro kanály systému zpráv” na stránce 928.

Sdílené obslužné rutiny připojení na MQCONNX

Pomocí sdílených manipulátorů připojení můžete sdílet manipulátory mezi různými podprocesy v rámci stejného procesu.

Zadáte-li sdílený manipulátor připojení, manipulátor připojení vrácený z volání MQCONNX může být předán v následných voláních MQI v libovolném podprocesu v procesu.

Poznámka: Pomocí manipulátoru sdíleného připojení v systému IBM MQ MQI client se můžete připojit ke správci front serveru, který nepodporuje sdílené manipulátory připojení.

Další informace viz “Použití MQCONNX” na stránce 887.

Sestavení aplikací pro IBM MQ MQI clients

Aplikace lze sestavit a spustit v prostředí IBM MQ MQI client . Aplikace musí být sestavena a propojena s použitými IBM MQ MQI client . Způsob, jakým jsou aplikace sestavovány a propojeny, se liší podle použité platformy a programovacího jazyka.

Má-li být aplikace spuštěna v klientském prostředí, můžete ji zapsat v jazycích uvedených v následující tabulce:

Tabulka 133. Programovací jazyky podporované v klientských prostředích

Platforma klienta	C	C++	COBOL	pTAL	RPG, se	Visual Basic
AIX	Ano	Ano	Ano			
IBM i	Ano		Ano		Ano	
Linux	Ano	Ano	Ano			
Windows	Ano	Ano	Ano			Ano

Multi *Propojení aplikací jazyka C s kódem IBM MQ MQI client*

Po napsání aplikace IBM MQ , kterou chcete spustit na serveru IBM MQ MQI client, ji musíte propojit s kódem IBM MQ MQI client .

Aplikaci můžete propojit s kódem IBM MQ MQI client dvěma způsoby:

1. Přímo připojením aplikace ke správci front. V takovém případě musí být správce front ve stejném počítači jako vaše aplikace.
2. K souboru knihovny klienta, který vám poskytuje přístup ke správcům front na stejném nebo jiném počítači.

Produkt IBM MQ poskytuje soubor knihovny klienta pro každé prostředí:

AIX **AIX**

Knihovna libmqic.a pro nepodprocesové aplikace nebo knihovna libmqic_r.a pro podprocesové aplikace.

Linux **Linux**

Knihovna libmqic.so pro nepodprocesové aplikace nebo knihovna libmqic_r.so pro podprocesové aplikace.

IBM i **IBM i**

Vytvořte vazbu mezi klientskou aplikací a servisním programem klienta LIBMQIC pro aplikace bez podpory podprocesů nebo servisním programem LIBMQIC_R pro aplikace s podporou podprocesů.

Windows **Windows**

MQIC32.LIB.

ALW *Propojení aplikací C++ s kódem IBM MQ MQI client*

Můžete psát aplikace pro spuštění na klientovi v C + +. Metody sestavení se liší v závislosti na prostředí.

Chcete-li získat informace o tom, jak propojit aplikace C++, prohlédněte si téma Sestavení IBM MQ programů C++.

Úplné podrobnosti o všech aspektech použití C++ viz Použití C++ .

Multi *Propojení aplikací v jazyce COBOL s kódem IBM MQ MQI client*

Po napsání aplikace v jazyce COBOL, kterou chcete spustit na serveru IBM MQ MQI client, ji musíte propojit s příslušnou knihovnou.

Produkt IBM MQ poskytuje soubor knihovny klienta pro každé prostředí:

AIX **AIX**

Propojte svou nepodprocesovou aplikaci COBOL s knihovnou libmqicb.a nebo s podprocesovou aplikací COBOL s libmqicb_r.a.

IBM i **IBM i**

Svázat klientskou aplikaci COBOL se servisním programem AMQCSTUB pro aplikace bez podprocesů nebo servisní program AMQCSTUB_R pro aplikace s podporou podprocesů.

Windows **Windows**

Propojte kód aplikace s knihovnou MQICCBB pro 32bitový jazyk COBOL. IBM MQ MQI client for Windows nepodporuje 16bitový COBOL.

Windows *Propojení aplikací Visual Basic s kódem IBM MQ MQI client*

Aplikace Microsoft Visual Basic můžete propojit s kódem IBM MQ MQI client na systému Windows.

Deprecated

V produktu IBM MQ 9.0je podpora pro Microsoft Visual Basic 6.0 zamítnuta. Třídy IBM MQ pro .NET jsou doporučenými náhradními technologiemi. Další informace viz téma 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 souboru Visual Basic" na stránce 1018
Informace, které je třeba vzít v úvahu při kódování programů IBM MQ v Microsoft Visual Basic. Parametr Visual Basic je podporován pouze v systému Windows.

"Příprava programů Visual Basic v adresáři Windows" na stránce 986
Informace, které je třeba zvážit při používání programů Microsoft Visual Basic na systému Windows.

Spuštění aplikací v prostředí IBM MQ MQI client

Aplikaci IBM MQ můžete spustit jak v úplném prostředí IBM MQ , tak v prostředí IBM MQ MQI client , aniž byste změnili kód, za předpokladu, že jsou splněny určité podmínky.

Tyto podmínky jsou, že:

- Aplikace se nemusí souběžně připojovat k více než jednomu správci front.
- Před název správce front není ve volání `MQCONN` nebo `MQCONNX` uvedena hvězdička (*).
- Aplikace nemusí používat žádnou z výjimek uvedených v části Které aplikace jsou spuštěny na serveru IBM MQ MQI client?

Poznámka: Knihovny, které používáte v době linkování, určují prostředí, ve kterém musí být vaše aplikace spuštěna.

Při práci v prostředí IBM MQ MQI client nezapomeňte, že:

- Každá aplikace spuštěná v prostředí IBM MQ MQI client má vlastní připojení k serverům. Aplikace vytvoří jedno připojení k serveru pokaždé, když vydá volání `MQCONN` nebo `MQCONNX` .
- Aplikace odesílá a získává zprávy synchronně. To znamená čekání mezi časem, kdy je volání vydáno na klientovi, a návratem kódu dokončení a kódu příčiny v síti.
- Všechny převody dat provádí server, ale informace o přepsání konfigurovaného CCSID počítače viz také MQCCSID .

Připojení aplikací IBM MQ MQI client ke správcům front

Aplikace spuštěná v prostředí 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 IBM MQ vydá volání MQCONN nebo MQCONNX, klient identifikuje, jak se má vytvořit připojení. Je-li volání MQCONNX vydáno aplikací v klientu IBM MQ , knihovna klienta MQI vyhledává informace o kanálu klienta v následujícím pořadí:

1. Použití obsahu polí `ClientConnOffset` nebo `ClientConnPtr` struktury MQCNO (je-li zadána). Tato pole identifikují strukturu definice kanálu (MQCD), která má být použita jako definice kanálu

připojení klienta. Podrobnosti připojení lze přepsat pomocí uživatelské procedury před připojením. Další informace viz téma “Odkazování na definice připojení pomocí uživatelské procedury před připojením z úložiště” na stránce 960.

2. Je-li nastavena proměnná prostředí **MQSERVER** , použije se kanál, který definuje.
3. Pokud je definován soubor `mqclient.ini` a sekce Kanály obsahuje atribut **ServerConnectionParms** , použije se kanál, který definuje. Další informace viz IBM MQ MQI client konfigurační soubor, `mqclient.ini` a Sekce Kanály konfiguračního souboru klienta.
4. Jsou-li nastaveny proměnné prostředí **MQCHLLIB** a **MQCHLTAB** , použije se tabulka definic kanálů klienta, na kterou odkazují. Alternativně z IBM MQ 9.0poskytuje proměnná prostředí **MQCCDTURL** ekvivalentní schopnost nastavení kombinace proměnných prostředí **MQCHLLIB** a **MQCHLTAB** . Je-li nastavena hodnota **MQCCDTURL** , použije se tabulka definic kanálů klienta, na kterou odkazuje. Další informace viz URL přístup k tabulce CCDT.
5. Pokud je definován soubor `mqclient.ini` a sekce Kanály obsahuje atributy **ChannelDefinitionDirectory** a **ChannelDefinitionFile** , tyto atributy se použijí k vyhledání tabulky definic kanálů klienta. Další informace viz IBM MQ MQI client konfigurační soubor, `mqclient.ini` a Sekce Kanály konfiguračního souboru klienta.
6. Pokud nejsou proměnné prostředí nastaveny, klient vyhledá tabulku definic kanálů klienta s cestou a názvem, které jsou vytvořeny z atributu **DefaultPrefix** sekce AllQueueManagers v souboru `mqs.ini` . Další informace viz sekce AllQueueSprávci souboru mqs.ini.

   Pokud hledání tabulky definic kanálů klienta selže, klient použije následující cesty:

   - Linux AIX V systému AIX 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`
   - MQ Appliance V systému IBM MQ Appliance: *QMname*`_AMQCLCHL.TAB`. Zobrazují se pod položkou `mqbackup:// URI`.

První z voleb popsaných v předchozím seznamu (pomocí polí `ClientConnOffset` nebo `ClientConnPtr` modulu MQCNO) je podporován pouze voláním MQCONNX. Pokud aplikace používá MQCONN namísto MQCONNX, informace o kanálu se vyhledávají zbývajícími pěti způsoby v pořadí uvedeném v seznamu. Pokud klient nenajde informace o kanálu, volání MQCONN nebo MQCONNX se nezdaří.

Název kanálu (pro připojení klienta) musí odpovídat názvu kanálu připojení serveru definovanému na serveru, aby bylo volání MQCONN nebo MQCONNX úspěšné.

Související pojmy

Webový adresovatelný přístup k tabulce definic kanálů klienta

Související úlohy

Konfigurace připojení mezi serverem a klientem

Související odkazy

Tabulka definic kanálů klienta

MQCNO-Volby připojení

Připojení klientských aplikací ke správcům front pomocí proměnných prostředí

Informace o kanálu klienta lze poskytnout aplikaci, která je spuštěna v prostředí klienta, pomocí proměnných prostředí.

Aplikace spuštěná v prostředí IBM MQ MQI client se může připojit ke správci front pomocí následujících proměnných prostředí:

MQSERVER

Proměnná prostředí **MQSERVER** se používá k definování minimálního kanálu. **MQSERVER** uvádí umístění serveru IBM MQ a komunikační metodu, která se má použít.

MQCHLLIB
Proměnná prostředí **MQCHLLIB** určuje cestu k adresáři se souborem obsahujícím tabulku CCDT (Client Channel Definition Table). Soubor je vytvořen na serveru, ale lze jej zkopírovat na pracovní stanici IBM MQ MQI client .

MQCHLTAB
Proměnná prostředí **MQCHLTAB** určuje název souboru obsahujícího tabulku CCDT (Client Channel Definition Table).

V systému IBM MQ 9.0poskytuje proměnná prostředí **MQCCDTURL** ekvivalentní schopnost nastavení kombinace proměnných prostředí **MQCHLLIB** a **MQCHLTAB** . **MQCCDTURL** vám umožňuje poskytnout soubor, ftp nebo http URL jako jedinou hodnotu, ze které lze získat tabulku definic kanálů klienta. Další informace naleznete v tématu Přístup k tabulce definic kanálů klienta s možností adresování na webu.

Připojení aplikací klienta ke správcům front pomocí struktury MQCNO
Definici kanálu můžete určit ve struktuře definice kanálu (MQCD), která je dodávána s použitím struktury MQCNO volání MQCONNX.

Další informace naleznete v tématu Vytvoření kanálu připojení klienta na serveru IBM MQ MQI client pomocí MQCNO.

Připojení aplikací klienta ke správcům front pomocí tabulky definic kanálů klienta
Pokud použijete příkaz MQSC DEFINE CHANNEL, budou vámi uvedené podrobnosti umístěny 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. Pokud existuje více než jedna vhodná definice kanálu, volba kanálu je ovlivněna atributy váhy kanálu klienta (CLNTWGHT) a afinity připojení (AFFINITY).

Použití automatického opětovného připojení klienta
Klientské aplikace se mohou znovu připojit automaticky, aniž by bylo nutné psát další kód, a to konfigurací několika 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 naleznete v tématu Automatické opětovné připojení klienta.

Role tabulky definic kanálů klienta
Tabulka CCDT (Client Channel Definition Table) obsahuje definice kanálů připojení klienta. Je zvláště užitečné, pokud se vaše klientské aplikace budou muset připojit k řadě alternativních správců front.

Tabulka definic kanálů klienta se vytvoří při definování správce front. Stejný soubor může použít více než jeden klient IBM MQ .

Existuje řada způsobů, jak může klientská aplikace používat tabulky CCDT. CCDT lze zkopírovat do klientského počítače. CCDT můžete zkopírovat do umístění sdíleného více než jedním klientem. CCDT můžete zpřístupnit pro klienta jako sdílený soubor, zatímco zůstane umístěn na serveru.

V produktu IBM MQ 9.0může být tabulka CCDT hostována v centrálním umístění, které je přístupné prostřednictvím identifikátoru URI, což odstraňuje potřebu individuální aktualizace tabulky CCDT pro každého implementovaného klienta.

Související pojmy
Webový adresovatelný přístup k tabulce definic kanálů klienta
Související úlohy
Přístup k definicím kanálů připojení klienta

Související odkazy

Tabulka definic kanálů klienta

Skupiny správců front v tabulce CCDT

V tabulce CCDT (Client Channel Definition Table) můžete definovat sadu připojení jako *skupinu správců front*. Aplikaci můžete připojit ke správci front, který je součástí skupiny správců front. To lze provést předponou názvu správce front ve volání MQCONN nebo MQCONNX s hvězdičkou.

Můžete se rozhodnout definovat připojení k více než jednomu počítači serveru, protože:

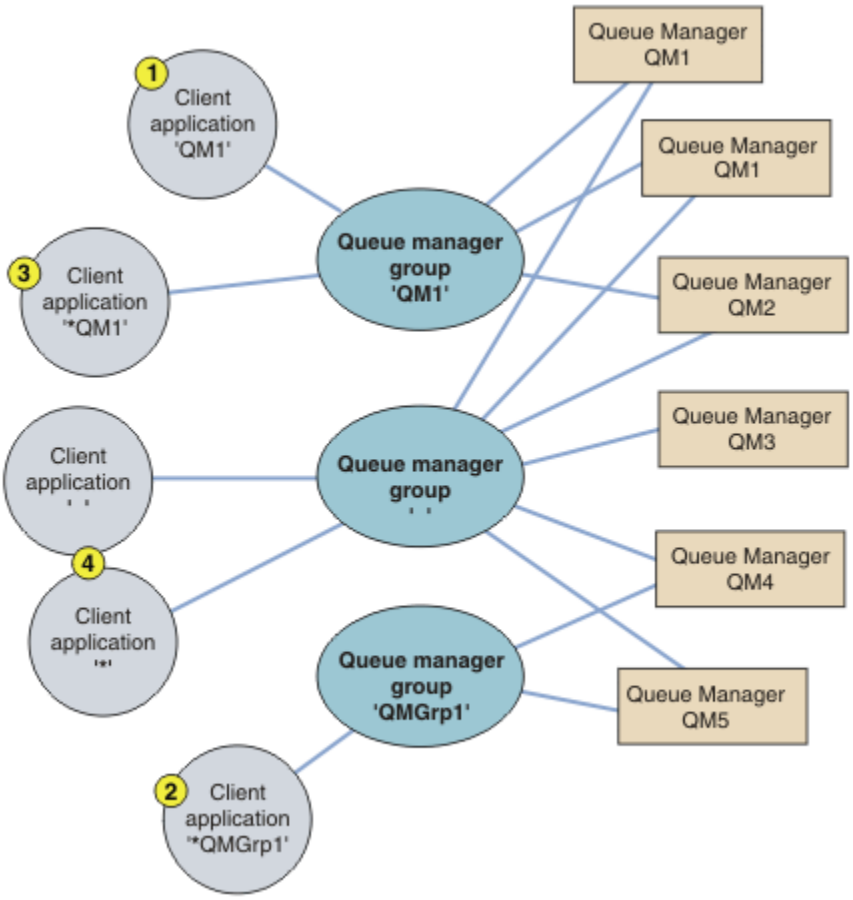
- Chcete připojit klienta k libovolné sadě spuštěných správců front za účelem zlepšení dostupnosti.
- Chcete znovu připojit klienta ke stejnému správci front, ke kterému byl naposledy úspěšně připojen, ale v případě selhání připojení se připojit k jinému správci front.
- Chcete-li mít možnost zopakovat pokus o připojení klienta k jinému správci front, dojde-li k selhání připojení, opětovným zadáním příkazu MQCONN v klientském programu.
- Chcete automaticky znovu připojit připojení klienta k jinému správci front v případě, že připojení selže, bez zápisu kódu klienta.
- Chcete automaticky znovu připojit připojení klienta k jiné instanci správce front s více instancemi, pokud to záložní instance převezme, bez zápisu kódu klienta.
- Chcete vyvážit připojení klientů v rámci více správců front, přičemž k některým správcům front se bude připojovat více klientů než k jiným.
- V případě, že velký objem připojení způsobí selhání, chcete opakované připojení mnoha připojení klienta rozložit na více správců front a v čase.
- Chcete mít možnost přesouvat správce front bez změny kódu klientské aplikace.
- Chcete psát klientské aplikační programy, které nepotřebují znát názvy správců front.

Není vždy vhodné připojit se k různým správcům front. Například rozšířený transakční klient nebo Java klient v produktu WebSphere Application Servermůže vyžadovat připojení k předvídatelné instanci správce front. Prostředí IBM MQ classes for Java nepodporuje automatické opětovné připojování klientů.

Skupina správců front je sada připojení definovaných v tabulce CCDT (Client Channel Definition Table). Sada je definována svými členy, kteří mají ve svých definicích kanálů stejnou hodnotu atributu **QMNAME** .

Obrázek 97 na stránce 893 je grafické znázornění tabulky připojení klienta zobrazující tři skupiny správců front, dvě pojmenované skupiny správců front zapsané v tabulce CCDT jako **QMNAME** (QM1) a **QMNAME** (QMGrp1)a jednu prázdnou nebo výchozí skupinu zapsanou jako **QMNAME** ('').

1. Skupina správců front QM1 má tři kanály připojení klienta, které se připojují ke správcům front QM1 a QM2. QM1 může být správce front pro více instancí umístěný na dvou různých serverech.
2. Výchozí skupina správců front má šest kanálů připojení klienta, které ji připojují ke všem správcům front.
3. QMGrp1 má kanály připojení klienta ke dvěma správcům front, QM4 a QM5.



Obrázek 97. Skupiny správců front

Čtyři příklady použití této tabulky připojení klienta jsou popsány pomocí očíslovaných klientských aplikací v souboru Obrázek 97 na stránce 893.

1. V prvním příkladu klientská aplikace předá název správce front QM1jako parametr **QmgrName** svému volání MQCONN nebo MQCONNX MQI. Kód klienta IBM MQ vybere odpovídající skupinu správců front QM1. Skupina obsahuje tři kanály připojení a konzola IBM MQ MQI client se pokouší připojit k serveru QM1 postupně pomocí každého z těchto kanálů, dokud nenalezne modul listener 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 možná připojení, tak v průběhu času, aby se rozložilo zatížení vytváření připojení.

   Volání MQCONN nebo MQCONNX vydané klientskou aplikací je úspěšné, když je ustanoveno připojení ke spuštěné instanci QM1.

2. Ve druhém příkladu klientská aplikace předá název správce front s předponou hvězdičky *QMGrp1 jako parametr **QmgrName** svému volání MQCONN nebo MQCONNX MQI. Klient IBM MQ vybere odpovídající skupinu správců front QMGrp1. Tato skupina obsahuje dva kanály připojení klienta a konzola IBM MQ MQI client se pokusí připojit k *libovolnému* správci front s použitím jednotlivých kanálů. V tomto příkladu musí produkt IBM MQ MQI client vytvořit úspěšné připojení; na názvu správce front, ke kterému se připojuje, nezáleží.

   Pravidlo pro pořadí provádění pokusů o připojení je stejné jako dříve. Jediný rozdíl spočívá v tom, že klient přidáním předpony názvu správce front s hvězdičkou označuje, že název správce front není relevantní.

   Volání MQCONN nebo MQCONNX vydané klientskou aplikací je úspěšné při vytvoření připojení ke spuštěné instanci libovolného správce front připojeného kanály ve skupině správců front QMGrp1 .

3. Třetí příklad je v podstatě stejný jako druhý, protože před parametr **QmgrName** je uvedena hvězdička *QM1. Příklad ukazuje, že nelze určit, ke kterému správci front se bude připojení kanálu klienta připojovat, a to prostřednictvím kontroly atributu QMNAME v jedné definici kanálu. Skutečnost, že atribut **QMNAME** definice kanálu je QM1, nestačí k tomu, aby bylo vyžadováno připojení ke správci front s názvem QM1. Pokud vaše klientská aplikace předpony svého parametru **QmgrName** s hvězdičkou, pak každý správce front je možný cíl připojení.

   V tomto případě jsou volání MQCONN nebo MQCONNX vydaná klientskou aplikací úspěšná, když je ustanoveno připojení ke spuštěné instanci buď QM1 , nebo QM2.
4. Čtvrtý příklad ilustruje použití výchozí skupiny. V tomto případě klientská aplikace předá hvězdičku '*'nebo prázdnou hodnotu ''jako parametr **QmgrName** svému volání rozhraní MQI MQCONN nebo MQCONNX . Podle konvence v definici kanálu klienta prázdný atribut **QMNAME** označuje výchozí skupinu správců front a prázdný parametr nebo hvězdička **QmgrName** odpovídá prázdnému atributu **QMNAME** .

   V tomto příkladu má výchozí skupina správců front připojení kanálu klienta ke všem správcům front. Po výběru výchozí skupiny správců front může být aplikace připojena k libovolnému správci front ve skupině.

   Volání MQCONN nebo MQCONNX vydané klientskou aplikací je úspěšné při vytvoření připojení ke spuštěné instanci libovolného správce front.

   Poznámka: Výchozí skupina se liší od výchozího správce front, ačkoli aplikace používá pro připojení k výchozí skupině správců front nebo k výchozímu správci front prázdný parametr **QmgrName** . Koncept výchozí skupiny správců front je relevantní pouze pro klientskou aplikaci a výchozí správce front pro serverovou aplikaci.

Kanály připojení klienta definujte pouze v jednom správci front, včetně kanálů, které se připojují ke druhému nebo třetímu správci front. Nedefinujte je ve dvou správcích front a poté se pokuste sloučit dvě tabulky definic kanálů klienta. Klient má přístup pouze k jedné tabulce definic kanálů klienta.

Příklady

Znovu se podívejte na seznam příčin použití skupin správců front na začátku tématu. Jak skupina správců front poskytuje tyto možnosti?

Připojte se k libovolné sadě správců front.
: Definujte skupinu správců front s připojením ke všem správcům front v sadě a připojte se ke skupině pomocí parametru **QmgrName** s předponou hvězdičky.

Znovu se připojte ke stejnému správci front, ale připojte se k jinému správci front, není-li správce front připojený k poslednímu času k dispozici.
: 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, pokud je připojení přerušeno nebo dojde k selhání správce front.

V případě selhání připojení se automaticky znovu připojte k jinému správci front.
: Připojte se ke skupině správců front pomocí volby MQCONNX **MQCNO** MQCNO_RECONNECT.

Automaticky se znovu připojte k jiné instanci správce front s více instancemi.
: Proveďte totéž jako předchozí příklad. Chcete-li v tomto případě omezit skupinu správců front tak, aby se připojovala k instancím konkrétního správce front s více instancemi, definujte skupinu s připojením pouze k instancím správce front s více instancemi.

  Můžete také požádat klientskou aplikaci o zadání jejího volání MQCONN nebo MQCONNX MQI bez hvězdičky s předponou parametru **QmgrName** . Aplikace klienta se tak může připojit pouze k uvedenému správci front. Nakonec můžete nastavit volbu **MQCNO** na hodnotu MQCNO_RECONNECT_Q_MGR. Tato volba přijímá opětovná připojení ke stejnému správci front, který byl dříve připojen. Pomocí této hodnoty můžete také omezit opětovná připojení na stejnou instanci normálního správce front.

Vyvažovat připojení klientů mezi správci front, přičemž k některým správcům front je připojeno více klientů než k jiným.

Definujte skupinu správců front a nastavte atribut **CLNTWGHT** pro každou definici kanálu klienta tak, aby nerovnoměrně distribuovala připojení.

Po selhání připojení nebo správce front rozloží zátěž připojení klienta nerovnoměrně a rozloží ji v čase.

Proveďte totéž jako předchozí příklad. Produkt IBM MQ MQI client náhodně provede opětovná připojení ve všech správcích front a v průběhu času tato opětovná připojení rozloží.

Přesuňte správce front bez změny kódu klienta.

Tabulka CCDT izoluje klientskou aplikaci od umístění správce front. CCDT je datový soubor, který lze definovat na klientovi, číst ze sdíleného umístění nebo načíst z webového serveru. Další informace viz Tabulka definic kanálů klienta.

Napište aplikaci klienta, která nezná názvy správců front.

Použijte názvy skupin správců front a vytvořte konvenci pojmenování pro názvy skupin správců front, která je relevantní pro klientské aplikace ve vaší organizaci a odráží architekturu vašich řešení namísto pojmenování správců front.

z/OS *Připojení ke skupinám sdílení front*

Aplikaci můžete připojit ke správci front, který je součástí skupiny sdílení front. To lze provést pomocí názvu skupiny sdílení front namísto názvu správce front ve 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. Provede se kontrola, zda je správce front, ke kterému se modul listener připojuje, členem skupiny sdílení front.

Další informace o sdílených frontách naleznete v tématu Sdílené fronty a skupiny sdílení front.

Příklady váhy kanálů a afinity

Tyto příklady ilustrují výběr kanálů připojení klienta při použití nenulových kanálů `ClientChannelVáhy` .

Atributy kanálu `ClientChannelVáha` a `ConnectionAffinity` řídí způsob výběru kanálů připojení klienta, když je pro připojení k dispozici více než jeden vhodný kanál. Tyto kanály jsou konfigurovány pro připojení k různým správcům front s cílem zajistit vyšší dostupnost, vyrovnávání pracovní zátěže nebo obojí. Volání MQCONN, která by mohla vést k připojení k jednomu z několika správců front, musí před název správce front zadat hvězdičku, jak je popsáno v tématu: Příklady volání MQCONN: Příklad 1. Název správce front obsahuje hvězdičku (*).

Použitelné kandidátské kanály pro připojení jsou ty kanály, u kterých se atribut QMNAME shoduje s názvem správce front zadaným ve volání MQCONN. Pokud mají všechny použitelné kanály pro připojení hodnotu `ClientChannel` nula (výchozí), jsou 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 pseudonáhodný výběr kanálu, příklady ukazují posloupnost akcí, které se mohou stát spíše než to, co rozhodně bude.

Příklad 1. Výběr kanálů, je-li vlastnost ConnectionAffinity nastavena na hodnotu PREFERRED

Tento příklad ukazuje, jak produkt IBM MQ MQI client vybírá kanál z tabulky CCDT, kde je volba ConnectionAffinity nastavena na hodnotu PREFERRED.

V tomto příkladu několik klientských počítačů používá tabulku CCDT (Client Channel Definition Table) poskytovanou správcem front. CCDT zahrnuje kanály připojení klienta s následujícími atributy (zobrazenými 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)
```

Aplikace vydá příkaz MQCONN (*CORE)

Kanál A není kandidátem pro toto připojení, protože se atribut QMNAME neshoduje. Kanály B, C a D jsou označeny jako kandidáty a jsou uvedeny v pořadí podle jejich váhy. V tomto příkladu může být objednávka C, B, D. Klient se pokusí připojit ke správci front na adrese core2.ops.company.example. Název správce front na této adrese není kontrolován, protože volání MQCONN obsahovalo v názvu správce front hvězdičku.

Je důležité si uvědomit, že spolu s produktem `AFFINITY(PREFERRED)` při každém připojení tohoto konkrétního klientského počítače umístí kanály do stejného počátečního pořadí předvoleb. To platí i v případě, že připojení jsou z různých procesů nebo v různých časech.

V tomto příkladu nelze dosáhnout správce front na adrese core.2.ops.company.example . Klient se pokusí připojit k souboru core1.ops.company.example , protože kanál B je následující v pořadí podle předvolby. Kromě toho je kanál C degradován, aby se stal nejméně preferovaným.

Druhé volání MQCONN (*CORE) je vydáno stejnou aplikací. Kanál C byl degradován na předchozí připojení, takže nejpreferovanější kanál je nyní B. Toto připojení je vytvořeno k souboru core1.ops.company.example.

Druhý počítač, který sdílí stejnou tabulku definic kanálů klienta, umístí kanály do jiného počátečního pořadí předvoleb. Například D, B, C. Za normálních okolností, kdy jsou všechny kanály funkční, jsou aplikace na tomto počítači připojeny k souboru core3.ops.company.example , zatímco aplikace na prvním počítači jsou připojeny k souboru core2.ops.company.example. To umožňuje vyrovnávání pracovní zátěže velkého počtu klientů v rámci více správců front a zároveň umožňuje každému jednotlivému klientovi připojit se ke stejnému správci front, je-li k dispozici.

Příklad 2. Výběr kanálů, je-li vlastnost ConnectionAffinity nastavena na hodnotu NONE
Tento příklad ukazuje, jak produkt IBM MQ MQI client vybírá kanál z tabulky CCDT, kde je vlastnost ConnectionAffinity nastavena na hodnotu NONE.

V tomto příkladu několik klientů používá tabulku CCDT (Client Channel Definition Table) poskytovanou správcem front. CCDT zahrnuje kanály připojení klienta s následujícími atributy (zobrazenými 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 příkaz MQCONN (*CORE). Stejně jako v předchozím příkladu není kanál A zohledněn, protože se název QMNAME neshoduje. Kanál B, C nebo D jsou vybrány na základě jejich váhy s pravděpodobností 50%, 30% nebo 20%. V tomto příkladu lze vybrat kanál B. Není vytvořeno žádné trvalé pořadí předvoleb.

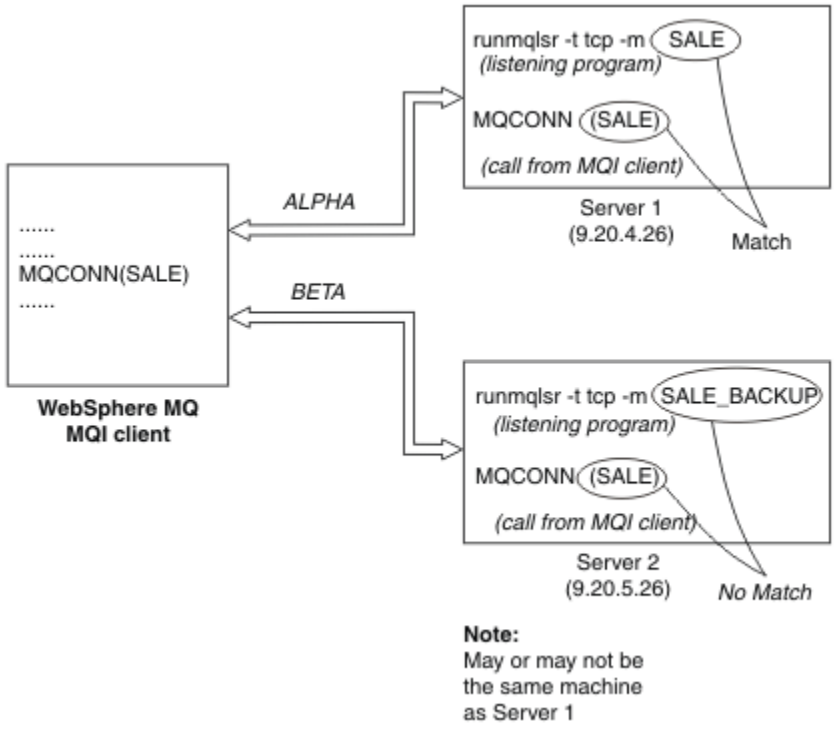
Provede se druhé volání MQCONN (*CORE). Opět je vybrán jeden ze tří použitelných kanálů se stejnou pravděpodobností. V tomto příkladu je zvolen kanál C. Avšak core2.ops.company.example neodpovídá, takže mezi zbývajícími kandidátními kanály je jiná volba. Je vybrán kanál B a aplikace je připojena k souboru core1.ops.company.example.

V případě volání AFFINITY (NONE) je každé volání MQCONN nezávislé na jakémkoli jiném volání. Proto když tato ukázková aplikace vytvoří třetí MQCONN (*CORE), může se ještě jednou pokusit o připojení přes přerušený kanál C, než zvolí jednu z možností B nebo D.

Příklady volání MQCONN
Příklady použití MQCONN pro připojení ke specifickému správci front nebo ke skupině správců front.

V každém z následujících příkladů je síť stejná; existuje připojení definované pro dva servery ze stejného serveru IBM MQ MQI client. (V těchto příkladech lze místo volání MQCONN použít volání MQCONNX.)

Na počítačích serveru jsou spuštěni dva správci front, jeden s názvem SALE a druhý s názvem SALE_BACKUP.



Obrázek 98. Příklad MQCONN

Definice kanálů v těchto příkladech jsou:

Definice PRODEJ:

```
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
Alfa	CLNTCONN	TCP	9.20.4.26	PRODEJ
BETA	CLNTCONN	TCP	9.20.5.26	PRODEJ

Co demonstrují příklady MQCONN
Příklady ukazují použití více správců front jako záložního systému.

Předpokládejme, že komunikační spojení se serverem 1 je dočasně přerušeno. Je demonstrováno použití více správců front jako záložního systému.

Každý příklad pokrývá jiné volání MQCONN a poskytuje vysvětlení toho, co se děje ve specifickém uvedeném příkladu, použitím následujících pravidel:

1. Tabulka CCDT (Client Channel Definition Table) je skenována v abecedním pořadí názvů kanálů pro název správce front (pole QMNAME) odpovídající názvu zadanému ve volání MQCONN.
2. Pokud je nalezena shoda, použije se definice kanálu.
3. Došlo k pokusu o spuštění kanálu do počítače identifikovaného názvem připojení (CONNAME). Pokud je tato akce úspěšná, aplikace pokračuje. Vyžaduje to:
   - Modul listener, který má být spuštěn na serveru.
   - Modul listener, který má být připojen ke stejnému správci front jako ten, ke kterému se chce klient 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), bude v souboru vyhledána další shoda. Pokud je nalezena shoda, zpracování pokračuje v kroku 1.
5. Pokud není nalezena žádná shoda nebo v tabulce definic kanálů klienta nejsou žádné další položky a kanál se nepodařilo spustit, aplikace se nemůže připojit. Ve volání MQCONN je vrácen příslušný kód příčiny a kód dokončení. Aplikace může provést akci na základě vrácených kódů příčiny a dokončení.

Příklad 1. Název správce front obsahuje hvězdičku (*)
V tomto příkladu není aplikace znepokojena tím, ke kterému správci front se připojuje. Aplikace zadá volání MQCONN pro název správce front včetně hvězdičky. Je zvolen vhodný kanál.

Aplikace vydává:

```
MQCONN (*SALE)
```

Podle pravidel se tak děje v této instanci:

1. V tabulce CCDT (Client Channel Definition Table) je vyhledán název správce front SALEodpovídající volání MQCONN aplikace.
2. Byly nalezeny definice kanálů ALPHA a BETA .
3. Pokud má jeden kanál hodnotu CLNTWGHT 0, je tento kanál vybrán. Pokud obě mají hodnotu CLNTWGHT 0, kanál ALPHA je vybrán, protože je první v abecedním pořadí. Pokud mají oba kanály nenulovou hodnotu CLNTWGHT, je jeden kanál náhodně vybrán na základě jeho váhy.
4. Došlo k pokusu o spuštění kanálu.
5. Pokud byl vybrán kanál BETA , je pokus o jeho spuštění úspěšný.
6. Pokud byl vybrán kanál ALPHA , pokus o jeho spuštění NENÍ úspěšný, protože komunikační spojení je přerušeno. Poté se použijí následující kroky:
   a. Jediný další kanál pro název správce front SALE je BETA.
   b. Byl proveden pokus o spuštění tohoto kanálu-tento pokus je úspěšný.
7. Kontrola, zda je modul listener spuštěn, ukazuje, že je spuštěn. Není připojen ke správci front SALE , ale protože parametr volání MQI obsahuje hvězdičku (*), není provedena žádná kontrola. Aplikace je připojena ke správci front SALE_BACKUP a pokračuje ve zpracování.

Příklad 2. Zadaný název správce front
V tomto příkladu se musí aplikace připojit ke konkrétnímu správci front. Aplikace zadá pro tento název správce front volání MQCONN. Je zvolen vhodný kanál.

Aplikace vyžaduje připojení ke specifickému správci front s názvem SALE, jak je patrné z volání MQI:

```
MQCONN (SALE)
```

Podle pravidel se tak děje v této instanci:

1. Tabulka CCDT (Client Channel Definition Table) je skenována v abecední posloupnosti názvů kanálů pro název správce front SALEodpovídající volání MQCONN aplikace.
2. První nalezená odpovídající definice kanálu je ALPHA.
3. Byl učiněn pokus o spuštění kanálu-tento pokus není úspěšný, protože komunikační spojení je přerušeno.
4. V tabulce definic kanálů klienta je znovu vyhledán název správce front SALE a byl nalezen název kanálu BETA .
5. Byl proveden pokus o spuštění kanálu-tento pokus byl úspěšný.
6. Kontrola, zda je spuštěn modul listener, ukazuje, že je spuštěn, ale není připojen ke správci front SALE .
7. V tabulce definic kanálů klienta nejsou žádné další položky. Aplikace nemůže pokračovat a obdrží návratový kód MQRC_Q_MGR_NOT_AVAILABLE.

Příklad 3. Název správce front je prázdný nebo hvězdička (*)
V tomto příkladu není aplikace znepokojena tím, ke kterému správci front se připojuje. Aplikace vydá příkaz MQCONN se zadáním prázdného názvu správce front nebo hvězdičky. Je zvolen vhodný kanál.

S tím se zachází stejným způsobem jako s “Příklad 1. Název správce front obsahuje hvězdičku (*)” na stránce 898.

Poznámka: Pokud by tato aplikace byla spuštěna v jiném prostředí než IBM MQ MQI clienta název by byl prázdný, pokusila by se připojit k výchozímu správci front. Nejedná se o případ, kdy je spuštěn z prostředí klienta; přistupovaným správcem front je ten, který je přidružen k modulu listener, k němuž se kanál připojuje.

Aplikace vydává:

```
MQCONN ("")
```

, nebo

```
MQCONN (*)
```

Podle pravidel se tak děje v této instanci:

1. Tabulka CCDT (Client Channel Definition Table) je prohledávána v abecední posloupnosti názvů kanálů, aby se zobrazil prázdný název správce front odpovídající 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 byl název správce front prázdný.
3. Další položka je pro název kanálu BETA.
4. Parametr `queue manager name` v definici je SALE. Tato hodnota se opět neshoduje s parametrem volání MQCONN, který vyžaduje, aby byl název správce front prázdný.
5. V tabulce definic kanálů klienta nejsou žádné další položky. Aplikace nemůže pokračovat a obdrží návratový kód MQRC_Q_MGR_NOT_AVAILABLE.

Spuštění v prostředí klienta

Zprávy odeslané IBM MQ aplikacemi spuštěnými na systému IBM MQ MQI clients přispívají ke spouštění přesně stejným způsobem jako všechny ostatní zprávy a lze je použít ke spouštění programů na serveru i na klientovi.

Spouštění je podrobně vysvětleno v tématu Spouštěcí kanály.

Monitor spouštěčů a aplikace, která se má spustit, musí být na stejném systému.

Výchozí charakteristiky spuštěné fronty jsou stejné jako v prostředí serveru. Pokud nejsou v klientské aplikaci zadány žádné volby řízení synchronizačního bodu MQPMO, které vkládají zprávy do spuštěné fronty, jež je lokální pro správce front z/OS , jsou zprávy vkládány v rámci pracovní jednotky. Pokud je pak

splněna spouštěcí podmínka, je zpráva spouštěče vložena do inicializační fronty v rámci stejné pracovní jednotky a nemůže být načtena monitorem spouštěče, dokud neskončí jednotka práce. Proces, který se má spustit, se nespustí, dokud neskončí jednotka práce.

Definice procesu
Musíte definovat definici procesu na serveru, protože je přidružena k frontě, na které je nastaveno spouštění.

Objekt procesu definuje, co se má spustit. Pokud klient a server nejsou spuštěny na stejné platformě, musí všechny procesy spuštěné monitorem spouštěčů definovat *ApplType*, jinak server použije své výchozí definice (tj. typ aplikace, která je obvykle přidružena k počítači serveru) a způsobí selhání.

Pokud je například monitor spouštěčů spuštěn v systému IBM MQ MQI client a chce odeslat požadavek na server v jiném operačním systému, musí být definována hodnota MQAT_WINDOWS_NT, jinak jiný operační systém použije své výchozí definice a proces selže.

monitor spouštěčů
Monitor spouštěčů poskytovaný jinými produkty nežz/OS IBM MQ se spouští v klientských prostředích pro systémy IBM i IBM i, AIX, 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)
```

- ALW Na platformách AIX, Linux, and Windows :

```
runmqtmc [-m QMgrName] [-q InitQ]
```

Výchozí inicializační fronta je SYSTEM.DEFAULT.INITIATION.QUEUE ve výchozím správci front. Inicializační fronta je místo, kde monitor spouštěčů hledá zprávy spouštěče. Poté volá programy pro příslušné spouštěcí zprávy. Tento monitor spouštěčů podporuje výchozí typ aplikace a je stejný jako `runmqtrm` s tím rozdílem, že propojuje knihovny klienta.

Příkazový řetězec sestavený monitorem spouštěčů je následující:

1. *ApplicId* z příslušné definice procesu. *ApplicId* je název programu, který se má spustit, jako by byl zadán na příkazovém řádku.
2. Struktura MQTMC2 , uzavřená v uvozovkách, získaná z inicializační fronty. Spustí se příkazový řetězec, který má tento řetězec přesně tak, jak je uveden, v uvozovkách, aby jej systémový příkaz přijal jako jeden parametr.
3. *EnvrData* z příslušné definice procesu.

Monitor spouštěčů se nepodívá, zda se v inicializační frontě nenachází další zpráva, dokud se nedokončí aplikace, kterou spustil. Pokud má aplikace mnoho zpracování, nemusí monitor spouštěčů držet krok s počtem příchozích zpráv spouštěče. Existují dva způsoby, jak se s touto situací vypořádat:

1. Mít spuštěno více monitorů spouštěčů

   Zvolíte-li spuštění více monitorů spouštěčů, můžete řídit maximální počet aplikací, které lze spustit najednou.

2. Spustit spuštěné aplikace na pozadí

   Pokud se rozhodnete spouštět aplikace na pozadí, produkt IBM MQ nebude omezovat počet aplikací, které lze spustit.

Chcete-li spustit spuštěnou aplikaci na pozadí v systémech AIX and Linux , musíte na konec souboru *EnvrData* definice procesu umístit znak & (ampersand).

CICS aplikace (jiné nežz/OS)
Jiný aplikační program nežz/OS CICS , který vydává volání MQCONN nebo MQCONNX , musí být definován pro CEDA jako RESIDENT. Pokud znovu propojíte serverovou aplikaci CICS jako klienta, riskujete ztrátu podpory synchronizačních bodů.

Jiný aplikační program nežz/OS CICS , který vydává volání MQCONN nebo MQCONNX , musí být definován pro CEDA jako RESIDENT. Chcete-li nastavit rezidentní kód na co nejmenší hodnotu, můžete se odkázat na samostatný program a zadat volání MQCONN nebo MQCONNX .

Je-li k definování připojení klienta použita proměnná prostředí MQSERVER, musí být určena v prostředí CICSENV.CMD .

Aplikace IBM MQ lze spouštět v prostředí serveru IBM MQ nebo v klientu IBM MQ bez změny kódu. Avšak v prostředí serveru IBM MQ může produkt CICS fungovat jako koordinátor synchronizačních bodů a vy použijete EXEC CICS SYNCPOINT a EXEC CICS SYNCPOINT ROLLBACK spíše než **MQCMIT** a **MQBACK**. Pokud je aplikace CICS jednoduše znovu propojena jako klient, podpora synchronizačního bodu je ztracena. **MQCMIT** a **MQBACK** musí být použity pro aplikaci spuštěnou na serveru IBM MQ MQI client.

ALW Příprava a spuštění aplikací CICS a Tuxedo

Chcete-li spouštět aplikace CICS a Tuxedo jako klientské aplikace, použijte jiné knihovny než ty, které používáte se serverovými aplikacemi. ID uživatele, pod kterým je aplikace spuštěna, se také liší.

Chcete-li připravit aplikace CICS a Tuxedo ke spuštění jako aplikace IBM MQ MQI client , postupujte podle pokynů v části Konfigurace rozšířeného transakčního klienta.

Všimněte si však, že informace, které se konkrétně zabývají přípravou aplikací CICS a Tuxedo, včetně ukázkových programů dodávaných s produktem IBM MQ, předpokládají, že připravujete aplikace ke spuštění na serverovém systému IBM MQ . V důsledku toho se informace týkají pouze knihoven IBM MQ , které jsou určeny pro použití na serverovém systému. Při přípravě klientských aplikací musíte provést následující akce:

- Použijte příslušnou knihovnu klientského systému pro vazby jazyka, které vaše aplikace používá. Příklad:
  - Linux AIX Pro aplikace napsané v jazyce C v systému AIX and Linuxpoužijte místo knihovny libmqm knihovnu libmqic.
  - Windows V systémech Windows použijte místo knihovny mqm.libknihovnu mqic.lib .
- Namísto knihoven systému serveru uvedených v Tabulka 134 na stránce 901 a Tabulka 135 na stránce 901použijte ekvivalentní knihovny systému klienta. Pokud v těchto tabulkách není uvedena systémová knihovna serveru, použijte stejnou knihovnu v klientském systému.

Tabulka 134. Systémové knihovny klienta na systému AIX and Linux

Knihovna pro serverový systém IBM MQ	**Ekvivalentní knihovna pro použití v klientském systému IBM MQ**
libmqmxa	libmqcxa

Tabulka 135. Knihovny klientských systémů na systémech Windows

Knihovna pro serverový systém IBM MQ	**Ekvivalentní knihovna pro použití v klientském systému IBM MQ**
mqmxa.lib	mqcxa.lib
mqmtux.lib	mqcxa.lib
mqmenc.lib	mqcxa.lib
mqmcics4.lib	mqccics4.lib

ID uživatele používané klientskou aplikací

Když spustíte serverovou aplikaci IBM MQ v adresáři CICS, obvykle se přepne z uživatele CICS na ID uživatele transakce. Když však spustíte aplikaci IBM MQ MQI client v adresáři CICS, zachová si oprávnění CICS .

ALW *Ukázkové programy CICS a Tuxedo*

Ukázkové programy CICS a Tuxedo pro použití na systémech AIX, Linux, and Windows .

Tabulka 136 na stránce 902 uvádí ukázkové programy CICS a Tuxedo, které jsou dodávány pro použití na klientských systémech AIX and Linux . Tabulka 137 na stránce 902 uvádí ekvivalentní informace pro klientské systémy Windows . Tabulky také uvádějí soubory, které se používají pro přípravu a spuštění programů. Popis ukázkových programů viz “Ukázka transakce CICS” na stránce 1044 a “Použití ukázek TUXEDO v systému AIX, Linux, and Windows” na stránce 1087.

Tabulka 136. Ukázkové programy pro klientské systémy AIX and Linux

Popis	Zdroj	Spustitelný modul
CICS PROGRAM	amqscic0.ccs	amqscicc
Hlavičkový soubor pro program CICS	amqscih0.h	-
Klientský program Tuxedo pro vložení zpráv	amqstxpx.c	-
Program klienta 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.flds	-
Zobrazit soubor s popisem pro programy Tuxedo	amqstxvx.v	-

Tabulka 137. Ukázkové programy pro klientské systémy Windows

Popis	Zdroj	Spustitelný modul
CICS transakce	amqscic0.ccs	amqscicc
Soubor záhlaví pro transakci CICS	amqscih0.h	-
Klientský program Tuxedo pro vložení zpráv	amqstxpx.c	-
Program klienta 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 s popisem pro programy Tuxedo	amqstxvx.v	-
Soubor Makefile pro programy Tuxedo	amqstxmc.mak	-
Soubor ENVFILE pro programy Tuxedo	amqstxen.env	-

ALW *Chybová zpráva AMQ5203, jak byla upravena pro aplikace CICS a Tuxedo*

Když spustíte aplikace CICS nebo Tuxedo, které používají rozšířeného transakčního klienta, můžete vidět standardní diagnostické zprávy. Jedna z těchto možností byla upravena pro použití s rozšířeným transakčním klientem.

Zprávy, které můžete vidět v souborech protokolu chyb IBM MQ , jsou zdokumentovány v části Diagnostické zprávy: AMQ4000-9999. Zpráva AMQ5203 byla upravena pro použití s rozšířeným transakčním klientem. Zde je text upravené zprávy:

AMQ5203: Vyskytla se chyba při volání rozhraní XA.

Vysvětlení

Číslo chyby je & 2, kde hodnota 1 označuje, že zadaná hodnota příznaků & 1 byla neplatná, hodnota 2 označuje, že ve stejném procesu došlo k pokusu o použití knihoven s podporou podprocesů a knihoven bez podpory podprocesů, hodnota 3 označuje, že došlo k chybě se zadaným názvem správce front '& 3', hodnota 4 označuje, že ID správce prostředků & 1 bylo neplatné, hodnota 5 označuje, že byl proveden pokus o použití druhého správce front s názvem '& 3 ', když byl již připojen jiný správce front, 6 označuje, že byl volán správce transakcí, když není aplikace připojena ke správci front, 7 označuje, že volání XA bylo provedeno, když probíhala jiná volání, 8 označuje, že řetězec xa_info' & 4 've volání xa_open obsahoval neplatnou hodnotu parametru pro název parametru' & 5 'a 9 označuje, že řetězec xa_info' & 4 've volání xa_open postrádá povinný parametr, název parametru' & 5 '.

Odezva uživatele

Opravte chybu a zopakujte operaci.

Windows Příprava a spuštění aplikací serveru Microsoft Transaction Server

Chcete-li připravit aplikaci MTS ke spuštění jako aplikace IBM MQ MQI client , postupujte podle těchto pokynů, které jsou vhodné pro vaše prostředí.

Obecné informace o vývoji aplikací Microsoft Transaction Server (MTS), které přistupují k prostředkům systému IBM MQ , naleznete v části MTS v Centru nápovědy IBM MQ .

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

- Pokud komponenta používá vazby jazyka C pro rozhraní MQI, postupujte podle pokynů v souboru “Příprava programů v jazyce C v adresáři Windows” na stránce 983 , ale propojte komponentu s knihovnou mqicxa.lib namísto mqic.lib.
- Pokud komponenta používá třídy IBM MQ C++, postupujte podle pokynů v části “Sestavení programů C++ na Windows” na stránce 535 , ale propojte komponentu s knihovnou imqx23vn.lib místo imqc23vn.lib.
- Pokud komponenta používá vazby jazyka Visual Basic pro rozhraní MQI, postupujte podle pokynů v části “Příprava programů Visual Basic v adresáři Windows” na stránce 986 , ale při definování projektu Visual Basic zadejte do pole **Argumenty podmíněné kompilace** hodnotu `MqType=3` .

Příprava a spuštění aplikací IBM MQ JMS

Aplikace IBM MQ JMS můžete spustit v režimu klienta s produktem WebSphere Application Server jako správcem transakcí. Mohou se zobrazit určité varovné zprávy.

Chcete-li připravit a spustit aplikace IBM MQ JMS v režimu klienta s produktem WebSphere Application Server jako správcem transakcí, postupujte podle pokynů v části “Použití IBM MQ classes for JMS/Jakarta Messaging” na stránce 78.

Při spuštění klientské aplikace IBM MQ JMS se mohou zobrazit následující varovné zprávy:

MQJE080

Nedostatečné jednotky licence-spustit setmqcap

MQJE081

Soubor obsahující informace o licenční jednotce je v nesprávném formátu-spusťte setmqcap

MQJE082

Soubor obsahující informace o jednotce licence nebyl nalezen-spusťte setmqcap

Uživatelské procedury, uživatelské procedury rozhraní API a instalovatelné služby 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 k rozšíření prostředků správce front naleznete v tématu Rozšíření prostředků správce front.

Informace o zápisu a kompilaci uživatelských procedur a instalovatelných služeb naleznete v dílčích tématech.

Související pojmy

Programy uživatelské procedury kanálu pro kanály MQI

Související odkazy

Odkaz uživatelské procedury rozhraní API

Referenční informace o rozhraní instalovatelných služeb

IBM i Referenční informace o rozhraní instalovatelných služeb na webu IBM i

ALW Zápis uživatelských procedur a instalovatelných služeb na webu AIX, Linux, and Windows

Můžete psát a kompilovat uživatelské procedury bez odkazování na jakékoli knihovny IBM MQ na systému AIX, Linux, and Windows.

Informace o této úloze

Toto téma platí pouze pro systémy AIX, Linux, and Windows . Podrobnosti o zápisu uživatelských procedur a instalovatelných služeb pro jiné platformy naleznete v příslušných tématech specifických pro danou platformu.

Je-li produkt IBM MQ nainstalován v jiném než výchozím umístění, musíte napsat a zkompilovat uživatelské procedury bez odkazování na knihovny produktu IBM MQ .

V systémech AIX, Linux, and Windows můžete psát a kompilovat uživatelské procedury, aniž byste propojili některou z těchto knihoven IBM MQ :

- mqmzf
- MQM
- mqmvx
- mqmvxd
- mqic
- mqutl

Existující uživatelské procedury, které jsou propojeny s těmito knihovnami, pokračují v práci za předpokladu, že na AIX and Linux systémech IBM MQ je nainstalován ve výchozím umístění.

Postup

1. Zahrňte soubor záhlaví cmqec.h .

   Zahrnutí tohoto souboru záhlaví automaticky zahrnuje soubory záhlaví cmqc.h, cmqxc.h a cmqzc.h .
2. Napište uživatelskou proceduru tak, aby byla volání MQI a DCI provedena prostřednictvím struktury MQIEP. Další informace o struktuře MQIEP viz Struktura MQIEP.
   - Instalovatelné služby
     - Pomocí parametru **Hconfig** můžete odkazovat na volání MQZEP.
     - Před použitím parametru **Hconfig** musíte zkontrolovat, zda se první 4 bajty **Hconfig** shodují s **StrucId** struktury MQIEP.
     - Další informace o zápisu instalovatelných komponent služby viz MQIEP.
   - Uživatelské procedury rozhraní API
     - Pomocí parametru **Hconfig** můžete odkazovat na volání MQXEP.

  - Před použitím parametru **Hconfig** musíte zkontrolovat, zda se první 4 bajty **Hconfig** shodují s **StrucId** struktury MQIEP.
  - Další informace o zápisu uživatelských procedur rozhraní API viz "Zápis uživatelských procedur rozhraní API" na stránce 921.
- Uživatelské procedury kanálu
  - Pomocí parametru **pEntryPoints** struktury MQCXP můžete odkazovat na volání MQI a DCI.
  - Před použitím produktu **pEntryPoints**je třeba zkontrolovat, zda je číslo verze MQCXP ve verzi 8 nebo vyšší.
  - Další informace o zápisu uživatelských procedur kanálu naleznete v části "Psaní programů uživatelské procedury kanálu" na stránce 931.
- Ukončení převodu dat
  - Pomocí parametru **pEntryPoints** struktury MQDXP můžete odkazovat na volání MQI a DCI.
  - Před použitím produktu **pEntryPoints**je třeba zkontrolovat, zda je číslo verze MQDXP ve verzi 2 nebo vyšší.
  - K vytvoření kódu pro převod dat, který používá parametr **pEntryPoints** , můžete použít příkaz **crtmqcvx** a zdrojový soubor amqsvfc0.c . Další informace jsou uvedeny v tématech "Zápis uživatelské procedury pro převod dat pro IBM MQ for Windows" na stránce 958 a "Zápis uživatelské procedury pro převod dat pro systémy IBM MQ for AIX or Linux" na stránce 955.
  - Máte-li existující uživatelské procedury pro převod dat, které byly vygenerovány pomocí příkazu **crtmqcvx** , musíte znovu vygenerovat uživatelské procedury pomocí aktualizovaného příkazu.
  - Další informace o zápisu uživatelských procedur pro převod dat viz "Zápis uživatelských procedur převodu dat" na stránce 950.
- Uživatelské procedury před připojením
  - Pomocí parametru **pEntryPoints** struktury MQNXP můžete 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 uživatelských procedur před připojením naleznete v části "Odkazování na definice připojení pomocí uživatelské procedury před připojením z úložiště" na stránce 960.
- Uživatelské procedury publikování
  - Pomocí parametru **pEntryPoints** struktury MQPSXP můžete odkazovat na volání MQI a DCI.
  - Před použitím produktu **pEntryPoints**je třeba zkontrolovat, zda je číslo verze MQPSXP ve verzi 2 nebo vyšší.
  - Další informace o zápisu uživatelských procedur publikování viz "Zápis a kompilace uživatelských procedur publikování" na stránce 961.
- Uživatelské procedury pracovní zátěže klastru
  - Pomocí parametru **pEntryPoints** struktury MQWXP můžete odkazovat na volání MQXCLWLN.
  - Před použitím produktu **pEntryPoints**je třeba zkontrolovat, zda je číslo verze MQWXP ve verzi 4 nebo vyšší.
  - Další informace o zápisu uživatelských procedur pracovní zátěže klastru viz "Zápis a kompilace uživatelských procedur pracovní zátěže klastru" na stránce 963.

Například v uživatelské proceduře kanálu volající MQPUT:

```
pChannelExitParms -> pEntryPoints -> MQPUT_Call(pChannelExitParms -> Hconn,
                                                Hobj,
                                                &md,
                                                &pmo,
                                                messlen,
                                                buffer,
                                                &CompCode,
                                                &Reason);
```

Další příklady jsou uvedeny v souboru "Použití ukázkových procedurálních programů IBM MQ" na stránce 1023.

3. Zkompilujte uživatelskou proceduru:
   - Neodkazujte na knihovny IBM MQ .
   - Nezahrnujte vloženou cestu RPath do žádné knihovny 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 923.
     - Uživatelské procedury kanálu, uživatelské procedury publikování, uživatelské procedury pracovní zátěže klastru: "Kompilace programů uživatelské procedury kanálu v systémech AIX, Linux, and Windows" na stránce 949.
     - Ukončení převodu dat: "Zápis uživatelských procedur převodu dat" na stránce 950.
4. Umístěte výstup na jedno z následujících míst:
   - Cesta dle vašeho výběru, kterou plně kvalifikujete při konfiguraci ukončení
   - Výchozí cesta k ukončení ve specifickém instalačním adresáři. Například *MQ_DATA_PATH*`/exits/installation2`.
   - Výchozí cesta ukončení

     Výchozí cesta k uživatelské proceduře je *MQ_DATA_PATH*`/exits` pro 32bitové uživatelské procedury a *MQ_DATA_PATH*`/exits64` pro 64bitové uživatelské procedury. Tyto cesty můžete změnit v souboru qm.ini nebo mqclient.ini . Další informace viz Cesta ukončení. V systémech Windows a Linuxmůžete změnit cestu pomocí Průzkumníka IBM MQ :

     a. Klepněte pravým tlačítkem myši na název správce front
     b. Klepněte na volbu **Vlastnosti ...**
     c. Klepněte na volbu **Uživatelské procedury** .
     d. V poli Výchozí cesta ukončení zadejte název cesty k adresáři, který obsahuje uživatelský program.

   Je-li uživatelská procedura umístěna jak do specifického instalačního adresáře, tak do výchozího adresáře cesty, použije specifickou uživatelskou proceduru instalačního adresáře instalace produktu IBM MQ uvedená v cestě. Například uživatelská procedura je umístěna v adresáři `/exits/installation2` a v adresáři `/exits`, nikoli však v adresáři `/exits/installation1`. IBM MQ Instalace `installation2` používá ukončení `/exits/installation2`. IBM MQ Instalace `installation1` používá uživatelskou proceduru z adresáře `/exits` .
5. V případě potřeby nakonfigurujte ukončení:
   - Instalovatelné služby: "Konfigurace služeb a komponent" na stránce 914.
   - Uživatelské procedury rozhraní API: "Konfigurace uživatelských procedur rozhraní API" na stránce 926.
   - Uživatelské procedury kanálu: "Konfigurace uživatelských procedur kanálu" na stránce 950.
   - Uživatelské procedury publikování: "Konfigurace uživatelských procedur publikování" na stránce 963.
   - Uživatelské procedury před připojením: PreConnect sekce konfiguračního souboru klienta.

ALW *Uživatelské procedury rozhraní API nepropojené s knihovnou MQI*

Za určitých okolností byste měli propojit existující uživatelskou proceduru rozhraní API, kterou nelze znovu kódovat pro použití ukazatelů funkce MQIEP, s knihovnou rozhraní API IBM MQ .

To je nezbytné, aby vaše stávající uživatelská procedura rozhraní API mohla být úspěšně načtena běhovým spojovacím programem vašeho systému do programů, které ještě nemají načteny ukazatele funkce.

Poznámka: Tyto informace jsou omezeny na existující uživatelské procedury rozhraní API, které přímo volají rozhraní MQI. To znamená, že uživatelské procedury, které nepoužívají, MQIEP. Je-li to možné, měli byste naplánovat změnu kódu uživatelské procedury tak, aby místo toho používala vstupní body MQIEP.

V produktu IBM MQ 8.0je **runmqsc** příkladem programu, který neodkazuje přímo na knihovnu MQI.

Proto se uživatelské proceduře rozhraní API, která nebyla propojena s požadovanou knihovnou rozhraní IBM MQ API, nebo znovu kódovaná pro použití MQIEP, nezdařilo načíst do produktu **runmqsc**.

V protokolu chyb správce front se zobrazují chyby, například AMQ6175: `Systém nemohl dynamicky načíst sdílenou knihovnuspolu` s kvalifikovaným textem, například `undefined symbol: MQCONN`.

a AMQ7214: `Modul pro uživatelskou proceduru rozhraní API 'myexitname' nebyl načten`.

Související úlohy

“Zápis uživatelských procedur a instalovatelných služeb na webu AIX, Linux, and Windows” na stránce 904
Můžete psát a kompilovat uživatelské procedury bez odkazování na jakékoli knihovny IBM MQ na systému AIX, Linux, and Windows.

ALW Instalovatelné služby a komponenty pro AIX, Linux, and Windows

Tento oddíl uvádí instalovatelné služby a funkce a komponenty, které jsou k nim přidruženy. Rozhraní k těmto funkcím je zdokumentováno, abyste vy nebo dodavatelé softwaru mohli dodávat komponenty.

Hlavní důvody pro poskytování instalovatelných služeb IBM MQ jsou:

- Chcete-li vám poskytnout flexibilitu při výběru, zda použít komponenty poskytované produkty IBM MQ , nebo je nahradit či rozšířit o další komponenty.
- Chcete-li povolit účast dodavatelů, poskytnutím komponent, které mohou používat nové technologie, bez provedení interních změn v produktech IBM MQ .
- Umožnit IBM MQ využívat nové technologie rychleji a levněji, a tak poskytovat produkty dříve a za nižší ceny.

Instalovatelné služby a *komponenty služeb* jsou součástí struktury produktu IBM MQ . Ve středu této struktury je část správce front, která implementuje 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 sada funkcí implementovaných pomocí jedné nebo více *komponent služby*. Každá komponenta je vyvolána pomocí správně navrženého, veřejně dostupného rozhraní. To umožňuje nezávislým dodavatelům softwaru a dalším třetím stranám poskytovat instalovatelné komponenty k rozšíření nebo nahrazení komponent poskytovaných produkty IBM MQ . Tabulka 138 na stránce 908 shrnuje služby a komponenty, které lze použít.

Tabulka 138. Souhrn instalovatelných komponent služby			
instalovatelná služba	**Dodávaná 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.	(Jsou předpokládána příslušná zařízení pro autorizaci platformy)
služba názvů	Není	Poskytuje podporu správci front pro vyhledání názvu správce front, který vlastní určenou frontu. • Definované uživatelem	• Správce jmen zapsaný třetí stranou nebo uživatelem

Rozhraní instalovatelných služeb je popsáno v tématu Referenční informace o rozhraní instalovatelných služeb.

Související úlohy

Konfigurace instalovatelných služeb

Zápis komponenty služby

Tato sekce popisuje vztah mezi službami, komponentami, vstupními body a návratovými kódy.

Funkce a komponenty

Každá služba se skládá ze sady souvisejících funkcí. Například služba názvů obsahuje funkci pro:

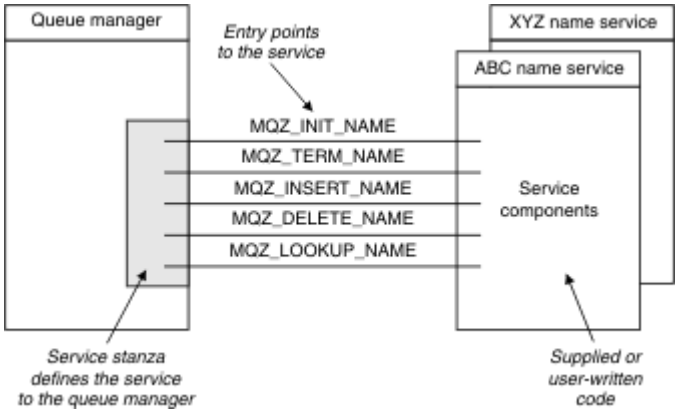
- Vyhledání názvu fronty a vrácení názvu správce front, kde je fronta definována
- Vložení názvu fronty do adresáře služby
- Odstranění názvu fronty z adresáře služby

Obsahuje také inicializační a ukončovací funkce.

Instalovatelná služba je poskytována jednou nebo více komponentami služby. Každá komponenta může provádět některé nebo všechny funkce definované pro tuto službu. Například v produktu IBM MQ for AIXdodaná komponenta služby autorizace, OAM, provádí všechny dostupné funkce. Další informace viz “Rozhraní autorizační služby” na stránce 911. Komponenta je také zodpovědná za správu všech základních prostředků nebo softwaru (například adresáře LDAP), které potřebuje k implementaci služby. Konfigurační soubory poskytují standardní způsob načtení komponenty a určení adres funkčních rutin, které poskytují.

Obrázek 99 na stránce 909 ukazuje, jak spolu služby a komponenty souvisí:

- Služba je definována pro správce front pomocí sekcí v konfiguračním souboru.
- Každá služba je podporována dodaným kódem ve správci front. Uživatelé nemohou tento kód změnit, a proto nemohou vytvářet své vlastní služby.
- Každá služba je implementována jednou nebo více komponentami; ty mohou být dodávány s produktem nebo napsány uživatelem. Pro službu lze vyvolat více komponent, z nichž každá podporuje různá zařízení v rámci služby.
- Vstupní body spojují komponenty služby s podpůrným kódem ve správci front.



Obrázek 99. Základní informace o službách, komponentách a vstupních bodech

Vstupní body

Každá komponenta služby je reprezentována seznamem adres vstupních bodů rutin, které podporují konkrétní instalovatelnou službu. Instalovatelná služba definuje funkci, kterou mají jednotlivé rutiny provádět.

Pořadí komponent služby při jejich konfiguraci definuje pořadí, ve kterém jsou vstupní body volány při pokusu o uspokojení požadavku na službu.

V zadaném souboru záhlaví `cmqzc.h` mají zadané vstupní body ke každé službě předponu `MQZID_`.

Pokud jsou služby přítomny, jsou služby načteny v předdefinovaném pořadí. V následujícím seznamu jsou uvedeny služby a pořadí jejich inicializace.

1. `NameService`
2. `AuthorizationService`
3. `UserIdentifierService`

`AuthorizationService` je jediná služba, která je standardně nakonfigurována. Nakonfigurujte `NameService` a `UserIdentifierService` ručně, chcete-li je použít.

Služby a komponenty služeb mají mapování jedna ku jedné nebo jedna ku mnoha. Pro každou službu lze definovat více komponent služby. Na systémech AIX and Linux musí hodnota služby sekce `ServiceComponent` odpovídat hodnotě názvu sekce služby v souboru qm.ini . V systému Windowsmusí hodnota klíče registru služeb `ServiceComponent` odpovídat hodnotě klíče registru názvů a je definována jako: `HKEY_LOCAL_MACHINE\SOFTWARE\IBM\WebSphere MQ\Installation\`*`MQ_INSTALLATION_NAME`*`\Configuration\QueueManager\`*`qmname`*`\` , kde *qmname* je název správce front.

V systémech AIX and Linux jsou komponenty služeb spouštěny v pořadí, v jakém jsou definovány v souboru qm.ini . V systému Windows, protože se používá registr Windows , IBM MQ vydá volání **RegEnumKey** , které vrátí hodnoty v abecedním pořadí. Proto se na systému Windows volají služby v abecedním pořadí, jak jsou definovány v registru.

Pořadí definic `ServiceComponent` je významné. Toto řazení určuje pořadí, ve kterém jsou komponenty spouštěny pro danou službu. Například `AuthorizationService` on Windows je nakonfigurován s výchozí komponentou OAM s názvem `MQSeries.WindowsNT.auth.service`. Pro tuto službu lze definovat další komponenty, které přepíší výchozí OAM. Není-li zadán parametr `MQCACF_SERVICE_COMPONENT` , použije se ke zpracování požadavku první zjištěná komponenta v abecedním pořadí a použije se název pro tuto komponentu.

Návratové kódy

Komponenty služeb poskytují správci front návratové kódy pro vytváření sestav za různých podmínek. Nahlásí úspěch nebo selhání operace a označí, zda má správce front pokračovat na další komponentu služby. Tuto indikaci nese samostatný parametr *Pokračování* .

Data komponenty

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 výhradní použití komponenty služby. Je sdílena všemi vyvoláními konkrétní funkce, a to i v případě, že jsou vytvořena z různých adresních prostorů nebo procesů. Je zaručeno, že bude adresovatelná z komponenty služby, kdykoli je volána. Velikost této oblasti musíte deklarovat v sekci *ServiceComponent* .

Inicializace a ukončení komponent
Použití voleb inicializace a ukončení komponenty.

Při vyvolání rutiny inicializace komponenty musí volat funkci správce front **MQZEP** pro každý vstupní bod podporovaný komponentou. **MQZEP** definuje vstupní bod služby. Předpokládá se, že všechny nedefinované výstupní body mají hodnotu NULL.

Komponenta je vždy vyvolána jednou s volbou primární inicializace, než je vyvolána jiným způsobem.

Komponentu lze vyvolat s volbou sekundární inicializace na určitých platformách. Může být například vyvolán jednou pro každý proces operačního systému, podproces nebo úlohu, pomocí které je služba přístupná.

Je-li použita sekundární inicializace:

- Komponentu lze pro sekundární inicializaci vyvolat více než jednou. Pro každé takové volání se vydá odpovídající volání pro sekundární ukončení, když služba již není potřebná.

  Pro služby pojmenování se jedná o volání MQZ_TERM_NAME.

  Pro autorizační služby se jedná o volání MQZ_TERM_AUTHORITY.
- Vstupní body musí být znovu zadány (voláním MQZEP) při každém volání komponenty pro primární a sekundární inicializaci.
- Pro komponentu se používá pouze jedna kopie dat komponenty; pro každou sekundární inicializaci neexistuje jiná kopie.
- Komponenta není vyvolána pro žádná další volání služby (z procesu operačního systému, podprocesu nebo úlohy, podle potřeby) před provedením sekundární inicializace.
- Komponenta musí nastavit parametr **Version** na stejnou hodnotu pro primární a sekundární inicializaci.

Komponenta je vždy vyvolána s volbou primárního ukončení jednou, když již není požadována. Pro tuto komponentu nejsou provedena žádná další volání.

Komponenta je vyvolána s volbou sekundárního ukončení, pokud byla vyvolána pro sekundární inicializaci.

správce oprávnění k objektu (OAM)
Komponenta služby autorizace dodávaná s produkty IBM MQ se nazývá OAM (Object Authority Manager).

Standardně je OAM aktivní a pracuje s řídicími příkazy **dspmqaut** (oprávnění k zobrazení), **dmpmqaut** (oprávnění k výpisu paměti) a **setmqaut** (oprávnění k nastavení nebo resetování).

Syntaxe těchto příkazů a jejich použití jsou popsány v tématu Administrace IBM MQ for Multiplatforms pomocí řídicích příkazů.

OAM pracuje s *entitou* činitele nebo skupiny:

- Linux AIX V systémech AIX and Linux je činitel ID uživatele nebo ID přidružené k aplikačnímu programu spuštěnému jménem uživatele; skupina je systémem definovaná kolekce činitelů.
- Windows Na systémech Windows je činitel ID uživatele Windows nebo ID přidružené k aplikačnímu programu spuštěnému jménem uživatele; skupina je skupina Windows .

Oprávnění lze udělit nebo odvolat na úrovni činitele nebo skupiny.

Při zadání požadavku MQI nebo zadání příkazu modul OAM zkontroluje, zda má entita přidružená k operaci oprávnění k provedení požadované operace a k přístupu k určeným prostředkům správce front.

Služba autorizace umožňuje rozšířit nebo nahradit kontrolu oprávnění poskytovanou pro správce front napsáním vlastní komponenty služby autorizace.

služba názvů
Služba názvů je instalovatelná služba, která poskytuje podporu správci front pro vyhledání názvu správce front, který vlastní určenou frontu. Ze služby názvů nelze načíst žádné další atributy fronty.

Služba názvů umožňuje aplikaci otevírat vzdálené fronty pro výstup, jako by se jednalo o lokální fronty. Služba názvů není vyvolána pro jiné objekty než fronty.

Poznámka: Vzdálené fronty musí mít atribut **Scope** nastaven na hodnotu CELL.

Když aplikace otevře frontu, hledá nejprve název fronty v adresáři správce front. Pokud ji tam nenajde, bude hledat v tolika službách názvů, kolik jich bylo nakonfigurováno, dokud nenajde takovou službu, která bude rozpoznávat název fronty. Pokud žádný název nerozpozná, otevření se nezdaří.

Služba názvů vrátí vlastnícího správce front pro tuto frontu. Správce front poté pokračuje s požadavkem MQOPEN, jako by příkaz zadal název fronty a správce front v původním požadavku.

Rozhraní služby názvů (NSI) je součástí rámce IBM MQ .

Jak služba názvů funguje

Pokud definice fronty určuje atribut **Scope** jako správce front, tj. SCOPE (QMGR) v MQSC, je definice fronty (spolu se všemi atributy fronty) uložena pouze v adresáři správce front. Tuto službu nelze nahradit instalovatelnou službou.

Pokud definice fronty určuje atribut **Scope** jako buňku, tj. SCOPE (CELL) v prostředí MQSC, bude definice fronty znovu uložena v adresáři správce front spolu se všemi atributy fronty. Název fronty a správce front jsou však uloženy také ve službě názvů. Pokud není k dispozici žádná služba, která by tyto informace mohla uložit, nelze definovat frontu s buňkou *Scope* .

Adresář, ve kterém jsou informace uloženy, může být spravován službou, nebo služba může pro tento účel použít základní službu, například adresář LDAP. V obou případech musí definice uložené v adresáři přetrvávat i po ukončení komponenty a správce front, dokud nejsou explicitně odstraněny.

Poznámka:

1. Chcete-li odeslat zprávu do lokální definice fronty vzdáleného hostitele (s oborem CELL) v jiném správci front v rámci buňky adresáře pojmenování, musíte definovat kanál.
2. Zprávy nelze získat 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, i když stále potřebujete přenosovou frontu na správce cílové fronty 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.

Oprávnění MQZ_CHECK_AUTHORITY
Kontroluje, zda má entita oprávnění k provedení jedné nebo více operací na uvedeném objektu.

MQZ_CHECK_PRIVILEGED
Zkontroluje, zda je zadaný uživatel privilegovaným uživatelem.

OPRÁVNĚNÍ MQZ_COPY_ALL_AUTHORITY
Zkopíruje všechna aktuální oprávnění, která existují pro odkazovaný objekt, do jiného objektu.

MQZ_DELETE_AUTHORITY
Odstraní všechny autorizace přidružené k určenému objektu.

MQZ_ENUMERATE_AUTHORITY_DATA
Načte všechna data oprávnění, která odpovídají uvedeným kritériím výběru.

MQZ_FREE_USER
Uvolní přidružené přidělené prostředky.

MQZ_GET_AUTHORITY
Získá oprávnění, které má entita pro přístup k uvedenému objektu.

Oprávnění MQZ_GET_EXPLICIT_AUTHORITY
Získá buď oprávnění, které má pojmenovaná skupina pro přístup k uvedenému objektu (ale bez dalšího oprávnění skupiny **nikdo**), nebo oprávnění, které má primární skupina jmenovaného činitele pro přístup k uvedenému objektu.

MQZ_INIT_AUTHORITY
Inicializuje komponentu služby autorizace.

DOTAZ_MQZ_INQUIRE
Dotáže se na podporované funkce autorizační služby.

MQZ_REFRESH_CACHE
Aktualizovat všechny autorizace.

MQZ_SET_AUTHORITY
Nastaví oprávnění, které má entita k uvedenému objektu.

MQZ_TERM_AUTHORITY
Ukončí komponentu služby autorizace.

Kromě toho v systému IBM MQ for Windowsposkytuje autorizační služba následující vstupní body 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).

Tyto názvy jsou definovány jako **typedef** v souboru záhlaví `cmqzc.h`, který lze použít k prototypu funkcí komponenty.

Inicializační funkce (**MQZ_INIT_AUTHORITY**) musí být hlavním vstupním bodem pro komponentu. Ostatní funkce jsou vyvolány prostřednictvím adresy vstupního bodu, kterou funkce inicializace přidala do vektoru vstupního bodu komponenty.

Rozhraní služby názvů
Služba názvů poskytuje vstupní body pro použití správcem front.

K dispozici jsou následující vstupní body:

MQZ_INIT_NAME
Inicializujte komponentu služby názvů.

MQZ_TERM_NAME
Ukončete komponentu služby názvů.

MQZ_LOOKUP_NAME
Vyhledejte název správce front pro určenou frontu.

MQZ_INSERT_NAME
Do adresáře používaného službou vložte položku obsahující název vlastnícího správce front pro určenou frontu.

MQZ_DELETE_NAME
Odstraňte položku pro uvedenou frontu z adresáře používaného službou.

Pokud je konfigurována více než jedna služba názvů:

- Pro vyhledávání je funkce MQZ_LOOKUP_NAME vyvolána pro každou službu v seznamu, dokud není název fronty vyřešen (pokud některá z komponent neindikuje, že má být vyhledávání zastaveno).
- Pro vložení je vyvolána funkce MQZ_INSERT_NAME pro první službu v seznamu, která podporuje tuto funkci.
- Pro odstranění je vyvolána funkce MQZ_DELETE_NAME pro první službu v seznamu, která podporuje tuto funkci.

Nemáte více než jednu komponentu, která podporuje funkce vložení a odstranění. Komponenta, která podporuje pouze vyhledávání, je však proveditelná a lze ji použít například jako poslední komponentu v seznamu k překladu libovolného názvu, který není znám žádné jiné komponentě služby názvů, na správce front, v němž lze název definovat.

V programovacím jazyku C jsou názvy definovány jako datové typy funkcí pomocí příkazu typedef. Ty lze použít k prototypu funkcí služby, aby se zajistilo, že parametry jsou správné.

Hlavičkový soubor, který obsahuje veškerý materiál specifický pro instalovatelné služby, je `cmqzc.h` pro jazyk C.

Kromě funkce inicializace (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 služby
Pro službu můžete nainstalovat více než jednu komponentu. To umožňuje komponentám poskytovat pouze dílčí implementace služby a spoléhat se na ostatní komponenty, aby poskytovaly zbývající funkce.

Příklad použití více komponent

Předpokládejme, že vytvoříte dvě komponenty služby názvů s názvem `ABC_name_serv` a `XYZ_name_serv`.

`ABC_name_serv`
Tato komponenta podporuje vložení názvu do adresáře služby nebo odstranění názvu z adresáře služby, ale nepodporuje vyhledávání názvu fronty.

`XYZ_name_serv`
Tato komponenta podporuje vyhledání názvu fronty, ale nepodporuje vložení názvu do adresáře služby nebo odstranění názvu z adresáře služby.

Komponenta `ABC_name_serv` obsahuje databázi názvů front a používá dva jednoduché algoritmy pro vložení nebo odstranění názvu z adresáře služeb.

Komponenta `XYZ_name_serv` používá jednoduchý algoritmus, který vrací pevný název správce front pro všechny názvy front, s nimiž je vyvolána. Nedrží databázi názvů front, a proto nepodporuje funkce vložení a odstranění.

Komponenty jsou nainstalovány ve stejném správci front. Sekce *`ServiceComponent`* jsou seřazeny tak, aby byla nejprve vyvolána komponenta `ABC_name_serv` . Jakákoli volání pro vložení nebo odstranění fronty v adresáři komponenty jsou zpracována komponentou `ABC_name_serv` ; je to jediný, kdo implementuje tyto funkce. Avšak volání vyhledání, které komponenta `ABC_name_serv` nemůže vyřešit, je předáno komponentě `XYZ_name_serv`pouze pro vyhledávání. Tato komponenta dodává název správce front ze svého jednoduchého algoritmu.

Vynechání vstupních bodů při použití více komponent

Pokud se rozhodnete použít k poskytnutí služby více komponent, můžete navrhnout komponentu služby, která neimplementuje určité funkce. Rámec instalovatelných služeb neklade žádná omezení, která byste mohli vynechat. V případě specifických instalovatelných služeb však může být vynechání jedné nebo více funkcí logicky nekonzistentní s účelem služby.

Příklad vstupních bodů používaných s více komponentami

Tabulka 139 na stránce 914 zobrazuje příklad instalovatelné služby názvů, pro kterou byly nainstalovány dvě komponenty. Každá z nich 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**) se předpokládá, že je NULL. Vstupní bod pro inicializaci je uveden v tabulce, ale není to nutné, protože inicializace je prováděna hlavním vstupním bodem komponenty.

Pokud má správce front použít instalovatelnou službu, použije vstupní body definované pro tuto službu (sloupce v části Tabulka 139 na stránce 914). Vezme-li se postupně každá komponenta, určí správce front adresu rutiny, která implementuje požadovanou funkci. Pak volá rutinu, pokud existuje. Je-li operace úspěšná, správce front použije všechny výsledky a informace o stavu.

Tabulka 139. Příklad vstupních bodů pro instalovatelnou službu

Číslo funkce	**Komponenta služby názvů ABC**	**Komponenta služby názvů XYZ**
MQZID_INIT_NAME (inicializováno)	ABC_initialize ()	XYZ_initialize ()
MQZID_TERM_NAME (ukončit)	ABC_terminate ()	XYZ_terminate ()
MQZID_INSERT_NAME (Vložit)	Vložení ABC_Insert ()	NEDEFINOVÁNO
MQZID_DELETE_NAME (Odstranit)	ABC_Delete ()	NEDEFINOVÁNO
MQZID_LOOKUP_NAME (Vyhledat)	NEDEFINOVÁNO	XYZ_Lookup ()

Pokud rutina neexistuje, správce front zopakuje tento proces pro další komponentu v seznamu. Kromě toho, pokud rutina existuje, ale vrátí kód označující, že operaci nemohla provést, bude pokus pokračovat s další dostupnou komponentou. Rutiny v komponentách služeb mohou vracet kód, který indikuje, že by neměly být prováděny žádné další pokusy o provedení operace.

Konfigurace služeb a komponent

Komponenty služeb konfigurujete pomocí konfiguračních souborů správce front, s výjimkou systémů Windows , kde má každý správce front v registru vlastní sekci.

Postup

1. Chcete-li definovat službu pro správce front a určit umístění modulu, přidejte do konfiguračního souboru správce front sekce `qm.ini`:
   - Každá použitá služba musí mít sekci `Service` , která definuje službu pro správce front. Další informace viz Sekce služby souboru qm.ini.
   - Pro každou komponentu v rámci služby musí existovat sekce `ServiceComponent` . Tato sekce identifikuje název a cestu k modulu, který obsahuje kód pro tuto komponentu. Další informace viz sekce ServiceComponent souboru qm.ini.

   Komponenta služby autorizace, známá jako OAM (Object Authority Manager), je dodávána s produktem. Při vytváření správce front je konfigurační soubor správce front (nebo registr v systémech Windows) automaticky aktualizován tak, aby obsahoval příslušné sekce pro autorizační službu a pro výchozí komponentu (OAM). Pro ostatní komponenty je třeba konfigurovat 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 to na platformě podporováno.
2. Zastavte a restartujte správce front, aby se komponenta aktivovala.

Související odkazy

Sekce služby souboru qm.ini

Sekce ServiceComponent souboru qm.ini

Aktualizace OAM po změně autorizace uživatele
V produktu IBM MQmůžete aktualizovat informace o skupině autorizace OAM ihned po změně členství uživatele ve skupině autorizace, což odráží změny provedené na úrovni operačního systému, aniž byste museli zastavit a restartovat správce front. Chcete-li tak učinit, zadejte příkaz **REFRESH SECURITY** .

Poznámka: Změníte-li oprávnění pomocí příkazu `setmqaut` , OAM tyto změny okamžitě implementuje.

Správci front ukládají data autorizace do lokální fronty s názvem SYSTEM.AUTH.DATA.QUEUE. Tato data jsou spravována produktem **amqzfuma.exe**.

Související odkazy
REFRESH SECURITY

IBM i Instalovatelné služby a komponenty na systému IBM i

Pomocí těchto informací získáte informace o instalovatelných službách a funkcích a komponentách, které jsou k nim přidruženy. Rozhraní k těmto funkcím je zdokumentováno, abyste vy nebo dodavatelé softwaru mohli dodávat komponenty.

Hlavní důvody pro poskytování instalovatelných služeb IBM MQ jsou:

- Chcete-li poskytnout flexibilitu při výběru, zda použít komponenty poskytované produktem IBM MQ for IBM i, nebo je nahradit či rozšířit o další komponenty.
- Chcete-li povolit účast dodavatelů, poskytnutím komponent, které mohou používat nové technologie, bez provedení interních změn v produktu IBM MQ for IBM i.
- Umožnit IBM MQ využívat nové technologie rychleji a levněji, a tak poskytovat produkty dříve a za nižší ceny.

Instalovatelné služby a *komponenty služeb* jsou součástí struktury produktu IBM MQ . Ve středu této struktury je část správce front, která implementuje 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 dostupná v produktu IBM MQ for IBM i je autorizační služba.

Každá instalovatelná služba je související sada funkcí implementovaných pomocí jedné nebo více *komponent služby*. Každá komponenta je vyvolána pomocí správně navrženého, veřejně dostupného rozhraní. To umožňuje nezávislým dodavatelům softwaru a dalším třetím stranám poskytovat instalovatelné komponenty k rozšíření nebo nahrazení komponent poskytovaných společností IBM MQ for IBM i. Tabulka 140 na stránce 915 shrnuje podporu pro autorizační službu.

Tabulka 140. Souhrn komponent služby autorizace

Dodávaná 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.	(Jsou předpokládána příslušná zařízení pro autorizaci platformy)
Komponenta služby názvů DCE **Poznámka:** DCE je podporováno pouze ve verzích produktu IBM MQ starších než V6.0.	• Umožňuje správcům front sdílet fronty, nebo • Definované uživatelem **Poznámka:** Sdílené fronty musí mít nastaven atribut **Scope** na hodnotu CELL.	• DCE je vyžadováno pro dodávanou komponentu, nebo • Správce jmen zapsaný třetí stranou nebo uživatelem

IBM i *Funkce a komponenty na systému IBM i*

Pomocí těchto informací můžete porozumět funkcím a komponentám, vstupním bodům, návratovým kódům a datům komponent, které můžete použít v produktu IBM MQ for IBM i.

Každá služba se skládá ze sady souvisejících funkcí. Například služba názvů obsahuje funkci pro:

- Vyhledání názvu fronty a vrácení názvu správce front, kde je fronta definována
- Vložení názvu fronty do adresáře služby
- Odstranění názvu fronty z adresáře služby

Obsahuje také inicializační a ukončovací funkce.

Instalovatelná služba je poskytována jednou nebo více komponentami služby. Každá komponenta může provádět některé nebo všechny funkce definované pro tuto službu. Komponenta je také zodpovědná za správu všech základních prostředků nebo softwaru, které potřebuje k implementaci služby. Konfigurační soubory poskytují standardní způsob načtení komponenty a určení adres funkčních rutin, které poskytují.

Služby a komponenty spolu souvisí takto:

- Služba je definována pro správce front pomocí sekcí v konfiguračním souboru.
- Každá služba je podporována dodaným kódem ve správci front. Uživatelé nemohou tento kód změnit, a proto nemohou vytvářet své vlastní služby.
- Každá služba je implementována jednou nebo více komponentami; ty mohou být dodávány s produktem nebo napsány uživatelem. Pro službu lze vyvolat více komponent, z nichž každá podporuje různá zařízení v rámci služby.
- Vstupní body spojují komponenty služby s podpůrným kódem ve správci front.

Vstupní body

Každá komponenta služby je reprezentována seznamem adres vstupních bodů rutin, které podporují konkrétní instalovatelnou službu. Instalovatelná služba definuje funkci, kterou mají jednotlivé rutiny provádět. Pořadí komponent služby při jejich konfiguraci definuje pořadí, ve kterém jsou vstupní body volány při pokusu o uspokojení požadavku na službu. V zadaném souboru záhlaví `cmqzc.h` mají zadané vstupní body ke každé službě předponu MQZID_.

Návratové kódy

Komponenty služeb poskytují správci front návratové kódy pro vytváření sestav za různých podmínek. Nahlásí úspěch nebo selhání operace a označí, zda má správce front pokračovat na další komponentu služby. Tuto indikaci nese samostatný parametr *Pokračování* .

Data komponenty

Jedna komponenta služby může vyžadovat sdílení dat mezi různými funkcemi. Instalovatelné služby poskytují volitelnou datovou oblast, která má být předána při každém vyvolání konkrétní komponenty služby. Tato datová oblast je určena pro výhradní použití komponenty služby. Je sdílena všemi vyvoláními dané funkce, i když jsou vytvořena z různých adresních prostorů nebo procesů. Je zaručeno, že bude adresovatelná z komponenty služby, kdykoli je volána. Velikost této oblasti musíte deklarovat v sekci *`ServiceComponent`* .

IBM i *Inicializace na IBM i*

Při vyvolání rutiny inicializace komponenty musí volat funkci MQZEP správce front pro každý vstupní bod podporovaný komponentou. MQZEP definuje vstupní bod ke službě. Předpokládá se, že všechny nedefinované výstupní body mají hodnotu NULL.

Primární inicializace
Komponenta je vždy vyvolána s touto volbou jednou, dříve než je vyvolána jiným způsobem.

Sekundární inicializace
Komponentu lze vyvolat pomocí této volby na určitých platformách. Může být například vyvolán jednou pro každý proces operačního systému, podproces nebo úlohu, pomocí které je služba přístupná.

Je-li použita sekundární inicializace:

- Komponentu lze pro sekundární inicializaci vyvolat více než jednou. Pro každé takové volání se vydá odpovídající volání pro sekundární ukončení, když služba již není potřebná.

  Pro autorizační služby se jedná o volání MQZ_TERM_AUTHORITY.
- Vstupní body musí být znovu zadány (voláním MQZEP) při každém volání komponenty pro primární a sekundární inicializaci.
- Pro komponentu se používá pouze jedna kopie dat komponenty; pro každou sekundární inicializaci neexistuje jiná kopie.
- Komponenta není vyvolána pro žádná další volání služby (z procesu operačního systému, podprocesu nebo úlohy, podle potřeby) před provedením sekundární inicializace.
- 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 požadována. Pro tuto komponentu nejsou provedena žádná další volání.

Sekundární ukončení
Komponenta je spuštěna s touto volbou, pokud byla spuštěna pro sekundární inicializaci.

IBM i *Konfigurace služeb a komponent na systému IBM i*

Komponenty služeb konfigurujete pomocí konfiguračních souborů správce front.

Postup

1. Chcete-li definovat službu pro správce front a určit umístění modulu, přidejte do konfiguračního souboru správce front sekce `qm.ini`:
   - Každá použitá služba musí mít sekci `Service` , která definuje službu pro správce front. Další informace viz Sekce služby souboru qm.ini.
   - Pro každou komponentu v rámci služby musí existovat sekce `ServiceComponent` . Tato sekce identifikuje název a cestu k modulu, který obsahuje kód pro tuto komponentu. Další informace viz sekce ServiceComponent souboru qm.ini.

   Komponenta služby autorizace, známá jako OAM (Object Authority Manager), je dodávána s produktem. Při vytváření správce front je konfigurační soubor správce front automaticky aktualizován tak, aby obsahoval příslušné sekce pro autorizační službu a pro výchozí komponentu (OAM). Pro ostatní komponenty je třeba konfigurovat 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 to na platformě podporováno.
2.

IBM i *Vytvoření vlastní komponenty služby v systému IBM i*

Pomocí těchto informací se dozvíte, jak vytvořit komponentu služby v systému IBM MQ for IBM i.

Chcete-li vytvořit vlastní komponentu služby, postupujte takto:

- Ujistěte se, že soubor záhlaví `cmqzc.h` je součástí vašeho programu.
- Vytvořte sdílenou knihovnu kompilací programu a jejím propojením se sdílenými knihovnami `libmqm*` a `libmqmzf*`.

Poznámka: Protože agent může být spuštěn v prostředí s podporou podprocesů, musíte sestavit OAM, aby se spustil v prostředí s podporou podprocesů. To zahrnuje použití verzí `libmqm` a `libmqmzfs` podporou podprocesů.

- Přidáním sekcí do konfiguračního souboru správce front definujte službu pro správce front a určete umístění modulu.
- Zastavte a restartujte správce front, aby se komponenta aktivovala.

IBM i *Autorizační služba na IBM i*

Autorizační služba je instalovatelná služba, která umožňuje správcům front vyvolat prostředky autorizace, například zkontrolovat, zda má ID uživatele oprávnění k otevření fronty.

Tato služba je součástí rozhraní SEI (security aktivační interface) systému IBM MQ , které je součástí rámce IBM MQ . Diskutovány jsou následující témata:

- “správce oprávnění k objektu (OAM)” na stránce 918
- “Definování služby pro operační systém” na stránce 918
- “Konfigurace sekcí autorizační služby” na stránce 918
- “Rozhraní služby autorizace na systému IBM i” na stránce 919

správce oprávnění k objektu (OAM)

Komponenta služby autorizace dodávaná s produkty IBM MQ se nazývá OAM (Object Authority Manager). Standardně je OAM aktivní a pracuje s následujícími řídicími příkazy:

- **WRKMQMAUT** pracovat s oprávněním
- **WRKMQMAUTD** pracovat s daty oprávnění
- **DSPMQMAUT** zobrazení oprávnění k objektu
- **GRTMQMAUT** udělit oprávnění k objektu
- **RVKMQMAUT** odvolání oprávnění k objektu
- **RFRMQMAUT** REFRESH SECURITY

Syntaxe těchto příkazů a jejich použití jsou popsány v nápovědě k CL příkazu. OAM pracuje s *entitou* činitele nebo skupiny.

Když je vydán požadavek MQI nebo příkaz, OAM zkontroluje autorizaci entity přidružené k operaci, aby zjistil, zda může provést následující akce:

- Proveďte požadovanou operaci.
- Přístup k určeným prostředkům správce front.

Služba autorizace umožňuje rozšířit nebo nahradit kontrolu oprávnění poskytovanou pro správce front napsáním vlastní komponenty služby autorizace.

Definování služby pro operační systém

Sekce služby autorizace v konfiguračním souboru správce front `qm.ini` definují službu autorizace pro správce front. Informace o typech sekcí viz “Konfigurace služeb a komponent na systému IBM i” na stránce 917 .

Konfigurace sekcí autorizační služby

V systému IBM MQ for IBM i:

Hlavní
Je profil uživatele systému IBM i .

Skupina
Jedná se o profil skupiny systému IBM i .

Autorizace lze udělit nebo odvolat pouze na úrovni skupiny. Požadavek na udělení nebo zrušení oprávnění uživatele aktualizuje primární skupinu pro tohoto uživatele.

Každý správce front má svůj vlastní konfigurační soubor správce front. Výchozí cesta a název souboru konfiguračního souboru správce front pro správce front QMNAME je například `/QIBM/UserData/mqm/qmgrs/QMNAME/qm.ini`.

Sekce *`Service`* a sekce *`ServiceComponent`* pro výchozí komponentu autorizace se přidají do produktu `qm.ini` automaticky, ale lze je přepsat pomocí `WRKENVVAR`. Všechny ostatní sekce *`ServiceComponent`* musí být přidány ručně.

Například následující sekce v konfiguračním souboru správce front definují dvě komponenty služby autorizace:

```
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 100. Sekce autorizační služby v qm.ini na IBM i

První sekce komponenty služby `MQ.UNIX.authorization.service` definuje výchozí komponentu služby autorizace, OAM. Pokud odeberete tuto sekci a restartujete správce front, modul OAM bude zakázán a nebudou provedeny žádné kontroly autorizace.

IBM i *Rozhraní služby autorizace na systému IBM i*

Rozhraní služby autorizace poskytuje několik vstupních bodů pro použití správcem front.

MQZ_AUTHENTICATE_USER
Ověřuje ID uživatele a heslo a může nastavit pole kontextu identity.

Oprávnění MQZ_CHECK_AUTHORITY
Kontroluje, zda má entita oprávnění k provedení jedné nebo více operací na uvedeném objektu.

OPRÁVNĚNÍ MQZ_COPY_ALL_AUTHORITY
Zkopíruje všechna aktuální oprávnění, která existují pro odkazovaný objekt, do jiného objektu.

MQZ_DELETE_AUTHORITY
Odstraní všechny autorizace přidružené k určenému objektu.

MQZ_ENUMERATE_AUTHORITY_DATA
Načte všechna data oprávnění, která odpovídají uvedeným kritériím výběru.

MQZ_FREE_USER
Uvolní přidružené přidělené prostředky.

MQZ_GET_AUTHORITY
Získá oprávnění, které má entita pro přístup k uvedenému objektu.

Oprávnění MQZ_GET_EXPLICIT_AUTHORITY
Získá buď oprávnění, které má pojmenovaná skupina pro přístup k uvedenému objektu (ale bez dalšího oprávnění skupiny **nikdo**), nebo oprávnění, které má primární skupina jmenovaného činitele pro přístup k uvedenému objektu.

MQZ_INIT_AUTHORITY
Inicializuje komponentu služby autorizace.

DOTAZ_MQZ_INQUIRE
Dotáže se na podporované funkce autorizační služby.

MQZ_REFRESH_CACHE
Aktualizovat všechny autorizace.

MQZ_SET_AUTHORITY
Nastaví oprávnění, které má entita k uvedenému objektu.

MQZ_TERM_AUTHORITY
Ukončí komponentu služby autorizace.

Tyto vstupní body podporují použití identifikátoru zabezpečení Windows (NT SID).

Tyto názvy jsou definovány jako **typedef** v souboru záhlaví `cmqzc.h`, který lze použít k prototypu funkcí komponenty.

Inicializační funkce (**MQZ_INIT_AUTHORITY**) musí být hlavním vstupním bodem pro komponentu. Ostatní funkce jsou vyvolány prostřednictvím adresy vstupního bodu, kterou funkce inicializace přidala do vektoru vstupního bodu komponenty.

Další informace viz "Vytvoření vlastní komponenty služby v systému IBM i" na stránce 917.

Multi Zápis a kompilace uživatelských procedur rozhraní API na platformě Multiplatforms

Uživatelské procedury rozhraní API umožňují psát kód, který mění chování volání rozhraní API produktu IBM MQ , například MQPUT a MQGET, a poté tento kód vložit bezprostředně před nebo bezprostředně po těchto voláních.

Poznámka: z/OS Není podporováno na systému IBM MQ for z/OS.

Proč používat uživatelské procedury rozhraní API?

Každá z vašich žádostí má určitou práci a její kód by měl tuto úlohu provádět co nejefektivněji. Na vyšší úrovni můžete použít standardy nebo obchodní procesy na konkrétního správce front pro všechny aplikace, které používají tohoto správce front. Je efektivnější dělat to nad úrovní jednotlivých aplikací, a tedy bez nutnosti měnit kód každé dotčené aplikace.

Zde je několik návrhů oblastí, ve kterých mohou být uživatelské procedury rozhraní API užitečné:

Zabezpečení
Z důvodu zabezpečení můžete provést ověření a zkontrolovat, zda jsou aplikace autorizovány pro přístup ke frontě nebo správci front. Můžete také policejní aplikace ' použití rozhraní API, ověření jednotlivých volání API, nebo dokonce parametry, které používají.

Flexibilita
Z důvodu flexibility můžete reagovat na rychlé změny ve svém obchodním prostředí, aniž byste měnili aplikace, které spoléhají na data v tomto prostředí. Můžete mít například uživatelské procedury rozhraní API, které reagují na změny v úrokových sazbách, směnných kurzech nebo ceně komponent ve výrobním prostředí.

Monitorování použití fronty nebo správce front
Chcete-li monitorovat použití fronty nebo správce front, můžete trasovat tok aplikací a zpráv, protokolovat chyby ve voláních rozhraní API, nastavit záznamy pro audit pro účely evidence nebo shromažďovat statistické údaje o využití pro účely plánování.

Co se stane, když se spustí uživatelská procedura rozhraní API?

Jakmile jste napsali uživatelský program a identifikovali jej v produktu IBM MQ, správce front automaticky vyvolá váš kód uživatelské procedury v registrovaných bodech.

Uživatelské procedury rozhraní API, které mají být spuštěny, jsou identifikovány v oddílech na platformě Multiplatforms. Toto téma pokrývá sekce v konfiguračních souborech `mqs.ini` a `qm.ini`.

Definice rutin se může vyskytovat na třech místech:

1. ApiExitCommon, v souboru mqs.ini , identifikuje rutiny pro celý IBM MQ, použité při spuštění správců front. Ty mohou být přepsány rutinami definovanými pro jednotlivé správce front (viz položka “3” na stránce 921 v tomto seznamu).
2. ApiExitŠablona v souboru mqs.ini identifikuje rutiny pro celý soubor IBM MQzkopírované do lokální sady ApiExit(viz položka “3” na stránce 921 v tomto seznamu) při vytvoření nového správce front.
3. ApiExitLokální v souboru qm.ini identifikuje rutiny, které se používají pro konkrétního správce front.

Při vytvoření nového správce front jsou definice šablon ApiExitv souboru mqs.ini zkopírovány do lokálních definic ApiExitv souboru qm.ini pro nového správce front. Při spuštění správce front se používají jak obecné definice ApiExit, tak lokální definice ApiExit. Lokální definice ApiExitnahrazují společné definice ApiExit, pokud obě identifikují rutinu se stejným názvem. Atribut Sequence , popsaný v části “Konfigurace uživatelských procedur rozhraní API” na stránce 926 , určuje pořadí, ve kterém rutiny definované v sekcích běží.

Použití uživatelských procedur rozhraní API v rámci více instalací produktu IBM MQ

Ujistěte se, že uživatelské procedury rozhraní API napsané pro dřívější verzi produktu IBM MQ jsou použity pro práci se všemi verzemi, protože změny provedené v uživatelských procedur v produktu IBM WebSphere MQ 7.1 nemusí fungovat se starší verzí. Další informace o změnách provedených v uživatelských procedur viz “Zápis uživatelských procedur a instalovatelných služeb na webu AIX, Linux, and Windows” na stránce 904.

Ukázky poskytnuté pro uživatelské procedury rozhraní API amqsaem a amqsaxe odrážejí změny požadované při zápisu uživatelských procedur. Klientská aplikace musí zajistit, aby byly před spuštěním aplikace propojeny správné knihovny IBM MQ , které odpovídají instalaci správce front, ke kterému je aplikace přidružena.

Multi *Zápis uživatelských procedur rozhraní API*

Můžete psát uživatelské procedury pro každé volání rozhraní API pomocí programovacího jazyka C.

Dostupné uživatelské procedury

Uživatelské procedury jsou k dispozici pro každé volání rozhraní API následujícím způsobem:

- MQCB, chcete-li znovu zaregistrovat zpětné volání pro určený popisovač objektu a řídit aktivaci a změny zpětného volání.
- MQCTL, chcete-li provádět řídicí akce na manipulátorů objektů otevřených pro připojení.
- MQCONN/MQCONNX-poskytnutí manipulátoru připojení správce front pro použití v následných voláních rozhraní API.
- MQDISC, chcete-li se odpojit od správce front
- MQBEGIN, chcete-li zahájit globální jednotku práce (UOW)
- MQBACK, pro vrácení jednotky UOW
- MQCMIT-potvrzení jednotky UOW
- MQOPEN-otevření prostředku 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-umístění zprávy do fronty.
- MQPUT, chcete-li umístit zprávu do fronty, která byla dříve otevřena pro přístup
- MQINQ, chcete-li zjistit atributy prostředku IBM MQ , který byl dříve otevřen pro přístup.
- MQSET-nastavení atributů fronty, která byla dříve otevřena pro přístup.
- MQSTAT-načtení informací o stavu
- MQSUB, chcete-li registrovat odběr aplikací pro konkrétní téma.

- MQSUBRQ-vytvoření požadavku na odběr

MQ_CALLBACK_EXIT poskytuje uživatelskou proceduru, která se má provést před a po zpracování zpětného volání. Další informace naleznete v tématu Zpětné volání-MQ_CALLBACK_EXIT.

Zápis uživatelských procedur rozhraní API

V rámci uživatelských procedur rozhraní API jsou volání v obecné podobě:

```
MQ_call_EXIT (parameters, context, ApiCallParameters)
```

kde *call* je název volání MQI bez předpony MQ ; například PUT, GET. *parameters* řídí funkci uživatelské procedury a primárně poskytuje komunikaci mezi uživatelskou procedurou a externími řídicími bloky MQAXP (struktura parametrů uživatelské procedury rozhraní API) a MQAXC (struktura kontextu uživatelské procedury rozhraní API). *context* popisuje kontext, ve kterém byla volána uživatelská procedura rozhraní API, a *ApiCallParameters* představuje parametry volání MQI.

Pro usnadnění zápisu uživatelské procedury rozhraní API je k dispozici ukázková uživatelská procedura amqsaxe0.c. Tato uživatelská procedura generuje položky trasování do souboru, který zadáte. Tuto ukázku můžete použít jako výchozí bod při psaní uživatelských procedur. Další informace o použití ukázkové uživatelské procedury viz “Ukázkový program uživatelské procedury rozhraní API” na stránce 1039.

Další informace o voláních uživatelské procedury rozhraní API, externích řídicích blocích a přidružených tématech naleznete v tématu Odkaz uživatelské procedury rozhraní API.

Obecné informace o tom, jak psát, kompilovat a konfigurovat uživatelskou proceduru, viz “Zápis uživatelských procedur a instalovatelných služeb na webu AIX, Linux, and Windows” na stránce 904.

Použití popisovačů zpráv v uživatelských rozhraních rozhraní API

Můžete řídit, ke kterým vlastnostem zprávy má uživatelská procedura rozhraní API přístup. Vlastnosti jsou přidruženy k obslužné rutiny ExitMsg. Vlastnosti nastavené v uživatelské proceduře vložení jsou nastaveny na vkládané zprávě, ale vlastnosti načtené v uživatelské proceduře získání nejsou vráceny aplikaci.

Při registraci funkce uživatelské procedury MQ_INIT_EXIT pomocí volání MQXEP MQI s hodnotou **Function** nastavenou na MQXF_INIT a **ExitReason** nastavenou na MQXR_CONNECTION předáte strukturu MQXEPO jako parametr **ExitOpts** . Struktura MQXEPO obsahuje pole ExitProperties , které určuje sadu vlastností, jež mají být zpřístupněny uživatelské proceduře. Je určen jako znakový řetězec představující předponu vlastností, která odpovídá názvu složky MQRFH2 .

Každá uživatelská procedura rozhraní API obdrží strukturu MQAXP obsahující pole manipulátoru ExitMsg. Toto pole je nastaveno na hodnotu generovanou produktem IBM MQ a je specifické pro připojení. Popisovač se proto nemění mezi rozhraními API stejného nebo různého typu na stejném připojení.

V uživatelské proceduře MQ_PUT_EXIT nebo MQ_PUT1_EXIT s hodnotou **ExitReason** MQXR_BEFORE, tj. v uživatelské proceduře rozhraní API provedené před vložením zprávy, jsou všechny vlastnosti (jiné než vlastnosti deskriptoru zpráv) přidružené k manipulátoru ExitMsgpo dokončení uživatelské procedury nastaveny na vkládanou zprávu. Chcete-li tomu zabránit, nastavte popisovač ExitMsgna hodnotu MQHM_NONE. Můžete také dodat jiný popisovač zprávy.

V MQ_GET_EXIT a MQ_CALLBACK_EXIT je popisovač ExitMsgvymazán z vlastností a naplněn vlastnostmi určenými v poli ExitProperties při registraci MQ_INIT_EXIT, kromě vlastností deskriptoru zprávy. Tyto vlastnosti nejsou zpřístupněny pro aplikaci získávání. Pokud aplikace provádějící získávání zadala v poli MQGMO (Volby získání zprávy) popisovač zprávy, budou uživatelské proceduře rozhraní API k dispozici všechny vlastnosti přidružené k tomuto popisovači, včetně vlastností deskriptoru zprávy. Chcete-li zabránit naplnění manipulátoru ExitMsgvlastnostmi, nastavte jej na hodnotu MQHM_NONE.

Poznámka: Pro vlastnosti zprávy ukončení, které se mají zpracovat v:

- Po funkci MQ_GET_EXIT musíte definovat funkci MQ_GET_EXIT před funkcí MQ_GET_EXIT pro ukončení.
- Před funkcí MQ_CALLBACK_EXIT musíte definovat funkci MQ_CB_EXIT pro uživatelskou proceduru.

V 9.3.2 V systému IBM MQ 9.3.2již neplatí předchozí příkazy týkající se MQ-GET-EXIT a MQ_CALLBACK_EXIT.

Ukázkový program amqsaem0.cje poskytován pro ilustraci použití popisovačů zpráv v uživatelských procedur rozhraní API.

Související odkazy

Uživatelské procedury, uživatelské procedury rozhraní API a odkaz na instalovatelné služby

Multi *Kompilace uživatelských procedur rozhraní API*

Po napsání uživatelské procedury ji zkompilujte a propojte následujícím způsobem.

Následující příklady ukazují příkazy použité pro ukázkový program popsaný v tématu “Ukázkový program uživatelské procedury rozhraní API” na stránce 1039. Pro jiné platformy než systémy Windows můžete najít ukázkový kód uživatelské procedury rozhraní API v produktu *MQ_INSTALLATION_PATH*/samp a kompilovanou a propojenou sdílenou knihovnu v produktu *MQ_INSTALLATION_PATH*/samp/bin.

Windows V případě systémů Windows můžete ukázkový kód uživatelské procedury rozhraní API najít v adresáři *MQ_INSTALLATION_PATH* \Tools\c\Samples. *MQ_INSTALLATION_PATH* představuje adresář, ve kterém byl nainstalován produkt IBM MQ .

Poznámka: Pokyny k programování 64bitových aplikací jsou uvedeny v části Standardy kódování na 64bitových platformách.

Pro klienty výběrového vysílání je třeba, aby uživatelské procedury rozhraní API a uživatelské procedury pro převod dat mohly být spuštěny na straně klienta, protože některé zprávy nemusí procházet správcem front. Následující knihovny jsou součástí balíků klienta i balíků serveru:

Tabulka 141. Knihovny, které jsou v balících klienta a serveru

Operační systém	Knihovny
AIX AIX	32bitový & 64bitový: libmqm.a & libmqm_r.a
IBM i IBM i	LIBMQM & LIBMQM_R
Linux Linux	32bitový & 64bitový: libmqm.so & libmqm_r.so
Windows Windows	32bitový & 64bitový: mqm.dll & mqm.pdb

Linux AIX *Kompilace uživatelských procedur rozhraní API na systémech AIX and Linux*

Příklady, jak kompilovat uživatelské procedury rozhraní API na systémech AIX and Linux .

Na všech platformách je vstupní bod modulu MQStart.

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

zapAIX

AIX

Kompilujte zdrojový kód uživatelské procedury rozhraní API zadáním jednoho z následujících příkazů:

32bitové aplikace

Bez podprocesů

```
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

Bez podprocesů

```
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
```

zapLinux

Linux

Kompilujte zdrojový kód uživatelské procedury rozhraní API zadáním jednoho z následujících příkazů:

31 bitové aplikace

Bez podprocesů

```
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

Bez podprocesů

```
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

Bez podprocesů

```
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
```

Windows *Kompilace uživatelských procedur rozhraní API na systémech Windows*

Zkompilujte a propojte ukázkový uživatelský program rozhraní API, `amqsaxe0.c`, na systému Windows .

Soubor typu manifest je volitelný dokument XML obsahující verzi nebo jiné informace, které lze vložit do kompilované aplikace nebo knihovny DLL.

Pokud takový dokument nemáte, vynechte parametr -manifest *manifest.file* v příkazu **mt** .

Upravte příkazy v příkladech v souboru Obrázek 101 na stránce 925 nebo Obrázek 102 na stránce 925 tak, aby se zkompilovaly a propojily `amqsaxe0.c` na Windows. 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 101. Kompilovat a propojit `amqsaxe0.c` na 32bitovém systému 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 102. Kompilovat a propojit `amqsaxe0.c` na 64bitovém systému Windows

Související pojmy

“Ukázkový program uživatelské procedury rozhraní API” na stránce 1039
Ukázková uživatelská procedura rozhraní API generuje trasování MQI do souboru určeného uživatelem s předponou definovanou v proměnné prostředí **MQAPI_TRACE_LOGFILE** .

IBM i *Kompilace uživatelských procedur rozhraní API na systému IBM i*
Kompilace uživatelských procedur rozhraní API v systému IBM i.

Uživatelská procedura je vytvořena následujícím způsobem (pro příklad jazyka C):

1. Vytvořte modul pomocí CRTCMOD. Zkompilujte jej tak, aby používal teraprostor, a to zahrnutím parametru `TERASPACE(*YES *TSIFC)`.
2. Vytvořte servisní program z modulu pomocí CRTSRVPGM. Musíte jej svázat s programem služby QMQM/LIBMQMZF_R pro uživatelské procedury rozhraní API s podporou podprocesů.

Konfigurace uživatelských procedur rozhraní API

Produkt IBM MQ nakonfigurujete tak, aby povoloval uživatelské procedury rozhraní API změnou informací o konfiguraci.

Chcete-li změnit informace o konfiguraci, musíte změnit sekce, které definují uživatelské procedury, a pořadí, ve kterém jsou spuštěny. Tyto informace lze změnit následujícími způsoby:

- Windows Linux Použití systémů IBM MQ Explorer na platformách Windows a Linux (x86 a x86-64).
- Windows Pomocí příkazu **amqmdain** na systému Windows.
- Multi Použití souborů `mqs.ini` a `qm.ini` přímo na platformě Multiplatforms.

  Soubor `mqs.ini` obsahuje informace týkající se všech správců front v konkrétním uzlu. Můžete jej najít v následujících umístěních:

  – Linux AIX V adresáři `/var/mqm` na systému AIX and Linux.

  – Windows V WorkPath uvedené v klíči `HKLM\SOFTWARE\IBM\WebSphere MQ` na systémech Windows .

  – IBM i V adresáři `/QIBM/UserData/mqm` na systému IBM i.

  Soubor `qm.ini` obsahuje informace týkající se specifického správce front. Pro každého správce front existuje jeden konfigurační soubor správce front, který je uložen v kořenovém adresáři adresářového stromu obsazeného správcem front. Například cesta a název konfiguračního souboru pro správce front s názvem QMNAME je:

  IBM i Na systémech IBM i:

  ```
  /QIBM/UserData/mqm/qmgrs/QMNAME/qm.ini
  ```

  Linux AIX Na systémech AIX and Linux:

  ```
  /var/mqm/qmgrs/QMNAME/qm.ini
  ```

  Windows Na systémech Windows:

  ```
  C:\ProgramData\IBM\MQ\qmgrs\QMNAME\qm.ini
  ```

  Před úpravou konfiguračního souboru jej zazálohujte, abyste měli kopii, ke které se můžete vrátit v případě potřeby.

  Konfigurační soubory můžete upravit jedním z následujících způsobů:

  – Automaticky pomocí příkazů, které mění konfiguraci správců front v uzlu.

  – Ručně, pomocí standardního textového editoru.

  Nastavíte-li nesprávnou hodnotu atributu konfiguračního souboru, hodnota se ignoruje a je vydána zpráva operátora, která označuje problém. Efekt je stejný, jako když atribut zcela chybí.

Sekce ke konfiguraci

Sekce, které se musí změnit, jsou následující:

ApiExitCommon

Definováno v adresáři `mqs.ini` a v adresáři IBM MQ Explorer na stránce vlastností IBM MQ v části Uživatelské procedury.

Při spuštění libovolného správce front jsou atributy v této sekci čteny a poté potlačeny pomocí uživatelských procedur rozhraní API definovaných v souboru `qm.ini`.

ApiExitTemplate

Definováno v adresáři `mqs.ini` a v adresáři IBM MQ Explorer na stránce vlastností IBM MQ v části Uživatelské procedury.

Při vytvoření libovolného správce front jsou atributy v této sekci zkopírovány do nově vytvořeného souboru qm.ini v lokální sekci ApiExit.

ApiExitLocal

Definováno v adresáři `qm.ini` a v adresáři IBM MQ Explorer na stránce vlastností správce front v části Uživatelské procedury.

Při spuštění správce front zde definované uživatelské procedury rozhraní API přepíší výchozí hodnoty definované v souboru `mqs.ini`.

Atributy pro sekce

- Pojmenujte uživatelskou proceduru rozhraní API pomocí následujícího atributu:

  Název = ApiExit_name
  Popisný název uživatelské procedury rozhraní API, která jí byla předána v poli ExitInfoNázev struktury MQAXP.

  Tento název musí být jedinečný, nesmí být delší než 48 znaků a musí obsahovat pouze platné znaky pro názvy objektů IBM MQ (například názvy front).

- Identifikujte modul a vstupní bod kódu uživatelské procedury rozhraní API, který se má spustit, pomocí následujících atributů:

  Funkce=název_funkce
  Název vstupního bodu funkce do modulu obsahujícího kód uživatelské procedury rozhraní API. Tento vstupní bod je funkcí MQ_INIT_EXIT.

  Délka pole je omezena hodnotou MQ_EXIT_NAME_LENGTH.

  Module=název_modulu
  Modul obsahující kód uživatelské procedury 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, musíte poskytnout verzi modulu uživatelské procedury rozhraní API bez podpory podprocesů i s podporou podprocesů. Verze s podprocesy musí mít příponu `_r` . Verze se podprocesy stubu aplikace IBM MQ implicitně připojí `_r` k danému názvu modulu před jeho načtením.

  Délka tohoto pole je omezena na maximální délku cesty, kterou platforma podporuje.

- Volitelně předejte data s uživatelskou procedurou pomocí následujícího atributu:

  Data=název_dat
  Data, která mají být předána uživatelské proceduře rozhraní API v poli ExitData struktury MQAXP.

  Pokud zahrnete tento atribut, počáteční a koncové mezery se odeberou, zbývající řetězec se ořízne na 32 znaků a výsledek se předá uživatelské proceduře. Vynecháte-li tento atribut, přednastavená hodnota 32 mezer se předá uživatelské proceduře.

  Maximální délka tohoto pole je 32 znaků.

- Identifikujte posloupnost této uživatelské procedury ve vztahu k jiným uživatelským procedurám pomocí následujícího atributu:

Sekvence=pořadové_číslo
Posloupnost, ve které je tato uživatelská procedura rozhraní API volána vzhledem k jiným uživatelským procedurám rozhraní API. Ukončení s nízkým pořadovým číslem je voláno 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 se správce front, která z nich má být volala jako první. Můžete určit, který byl volán po události, vložením času nebo značkovače do oblasti ExitChainoznačené ExitChainAreaPtr v MQAXP nebo zápisem vlastního souboru protokolu.

Tento atribut je číselná hodnota bez znaménka.

Ukázkové sekce

Ukázkový soubor mqs.ini obsahuje následující sekce:

ApiExitTemplate
Tato sekce definuje uživatelskou proceduru s popisným názvem `OurPayrollQueueAuditor`, názvem modulu `auditora` pořadovým číslem 2. Hodnota dat 123 je předána uživatelské proceduře.

ApiExitCommon
Tato sekce definuje uživatelskou proceduru s popisným názvem `MQPoliceman`, názvem modulu `tmqpa` pořadovým číslem 1. Předaná data 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 uživatelské procedury ApiExits popisným názvem `ClientApplicationAPIchecker`, názvem modulu `ClientAppCheckera` pořadovým číslem 3.

```
qm.ini

   ApiExitLocal:
      Name=ClientApplicationAPIchecker
      Sequence=3
      Function=EntryPoint
      Module=/usr/Dev/ClientAppChecker
      Data=9.20.176.20
```

Programy uživatelské procedury kanálu pro kanály systému zpráv

Tato kolekce témat obsahuje informace o programech uživatelské procedury kanálu IBM MQ pro kanály systému zpráv.

Agenti MCA (Message Channel Agent) mohou také volat uživatelské procedury pro převod dat. Další informace o zápisu uživatelských procedur pro převod dat viz “Zápis uživatelských procedur převodu dat” na stránce 950.

Některé z těchto informací se vztahují také na uživatelské procedury v kanálech MQI, které připojují produkt IBM MQ MQI clients ke správcům front. Další informace naleznete v tématu Programy uživatelské procedury kanálu pro kanály MQI.

Programy uživatelské procedury kanálu jsou volány na definovaných místech ve zpracování prováděném programy MCA.

Některé z těchto uživatelských programů pracují v komplementárních dvojicích. Je-li například program uživatelské procedury volán odesílajícím agentem MCA, aby šifroval zprávy pro přenos, musí doplňkový proces fungovat na přijímacím konci, aby mohl proces zvrátit.

Tabulka 142 na stránce 929 zobrazuje typy uživatelské procedury kanálu, které jsou k dispozici pro každý typ kanálu.

Tabulka 142. Pro každý typ kanálu jsou k dispozici uživatelské procedury kanálu.

Typ kanálu	Ukončení zprávy	Uživatelská procedura 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	
Odesílací kanál 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 vztahuje pouze na kanály odesilatele klastru a přijímací kanály klastru.

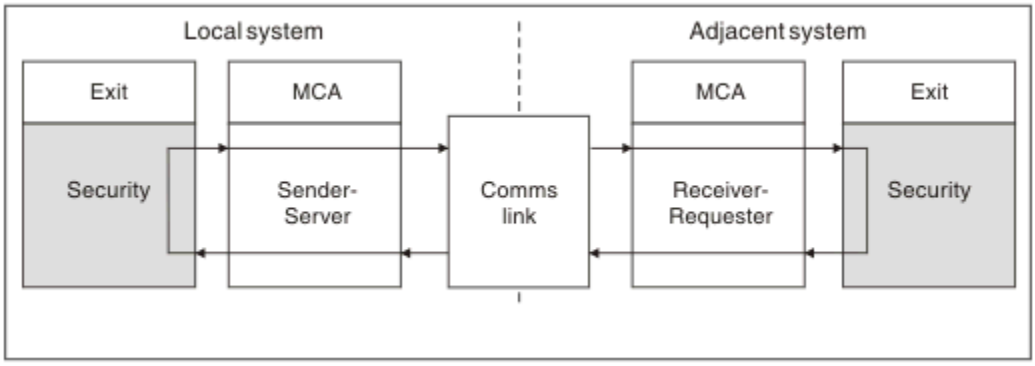
Pokud se chystáte spustit uživatelské procedury kanálu na klientovi, nemůžete použít proměnnou prostředí MQSERVER. Namísto toho vytvořte a odkažte na tabulku CCDT (Client Channel Definition Table), jak je popsáno v tématu Tabulka definic kanálů klienta.

Přehled zpracování

Přehled toho, jak MCA používají programy uživatelských procedur kanálu.

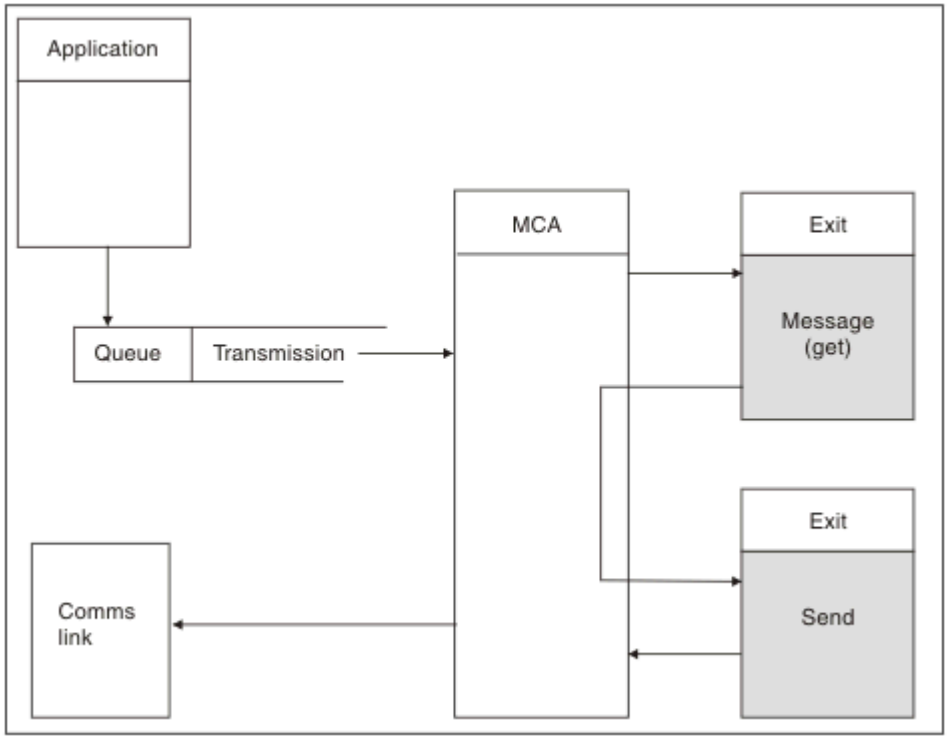
Při spuštění si MCA vyměňují spouštěcí dialogové okno pro synchronizaci zpracování. Pak se přepnou na výměnu dat, která zahrnuje uživatelské procedury zabezpečení. Tyto uživatelské procedury musí být úspěšně ukončeny, aby mohla být fáze spuštění dokončena a aby bylo možné přenášet zprávy.

Fáze kontroly zabezpečení je smyčka, jak ukazuje Obrázek 103 na stránce 930.

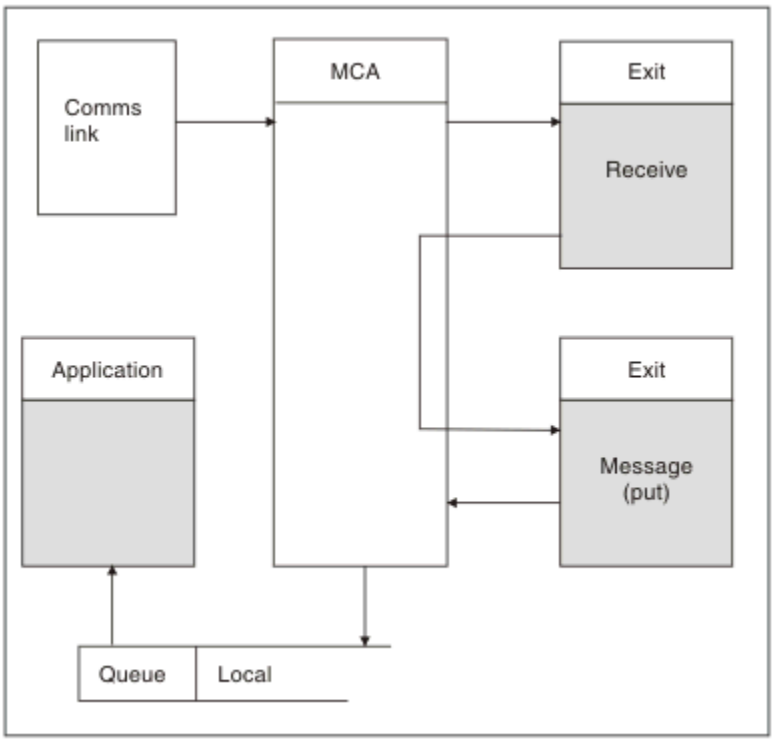


Obrázek 103. Smyčka uživatelské procedury zabezpečení

Během fáze přenosu zpráv odesílající agent MCA obdrží zprávy z přenosové fronty, zavolá uživatelskou proceduru pro zprávu, zavolá uživatelskou proceduru pro odeslání a poté odešle zprávu přijímající agentovi MCA, jak je uvedeno v části Obrázek 104 na stránce 930.



Obrázek 104. Příklad uživatelské procedury odeslání na odesílacím konci kanálu zpráv



Obrázek 105. Příklad uživatelské procedury příjmu na přijímacím konci kanálu zpráv

Přijímající agent MCA přijme zprávu z komunikačního propojení, zavolá uživatelskou proceduru pro příjem, zavolá uživatelskou proceduru pro zprávu a pak ji vloží do lokální fronty, jak ukazuje Obrázek 105 na stránce 931. (Před zavoláním uživatelské procedury pro příjem může být volána více než jednou.)

Psaní programů uživatelské procedury kanálu

Následující informace vám mohou pomoci při psaní programů pro uživatelské procedury kanálu.

Uživatelské procedury a programy uživatelských procedur kanálu mohou používat všechna volání MQI, s výjimkou případů uvedených v následujících sekcích. Pro produkt MQ V7 a novější struktura MQCXP verze 7 a vyšší obsahuje manipulátor připojení hConn, který lze použít namísto vydání MQCONN. Chcete-li získat manipulátor připojení pro starší verze, musí být vydán 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 mít zajištění neporušenosti vláken.

Pro uživatelské procedury v kanálech připojení klienta závisí správce front, ke kterému se uživatelská procedura pokouší připojit, na způsobu propojení uživatelské procedury. Pokud byla uživatelská procedura propojena s produktem MQM.LIB (nebo QMQM/LIBMQM v systému IBM i) a nezadáte název správce front ve volání MQCONN, uživatelská procedura se pokusí připojit k výchozímu správci front v systému. Pokud byla uživatelská procedura propojena s produktem MQM.LIB (nebo QMQM/LIBMQM v systému IBM i) a zadáte název správce front, který byl předán uživatelské proceduře prostřednictvím pole QMgrName MQCD, uživatelská procedura se pokusí připojit k tomuto správci front. Pokud byla uživatelská procedura propojena s produktem MQIC.LIB nebo jakákoli jiná knihovna, volání MQCONN selže bez ohledu na to, zda jste zadali název správce front.

Měli byste se vyvarovat změny stavu transakce přidružené k předanému hConn v uživatelské proceduře kanálu; nesmíte používat příkazová slova MQCMIT, MQBACK nebo MQDISC s kanálem hConna nemůžete použít příkaz MQBEGIN verb určující kanál hConn.

Je-li při použití příkazu MQCONNX zadána volba MQCNO_HANDLE_SHARE_BLOCK nebo volba MQCNO_HANDLE_SHARE_NO_NO_BLOCK pro vytvoření nového připojení k systému IBM MQ , je vaší povinností ověřit, zda je připojení správně spravováno a zda je od správce front správně odpojeno.

Například uživatelská procedura kanálu, která vytvoří nové připojení ke správci front při každém vyvolání bez odpojení, bude mít za následek sestavení manipulátorů připojení a zvýšení počtu podprocesů agenta.

Uživatelská procedura je spuštěna ve stejném podprocesu jako samotný agent MCA a používá stejný manipulátor připojení. Takže běží uvnitř stejné jednotky UOW jako MCA a všechna volání provedená v synchronizačním bodě jsou potvrzena nebo odvolána kanálem na konci dávky.

Uživatelská procedura zpráv kanálu proto může odesílat zprávy oznámení, které jsou do této fronty potvrzeny pouze v případě, že je dávka obsahující původní zprávu potvrzena. Je tedy možné zadat volání MQI synchronizačního bodu z uživatelské procedury pro zprávy kanálu.

Uživatelská procedura kanálu může změnit pole v MQCD. Tyto změny však nejsou provedeny, s výjimkou uvedených okolností. Pokud program uživatelské procedury kanálu změní pole v datové struktuře MQCD, proces kanálu IBM MQ novou hodnotu ignoruje. Nová hodnota však zůstává v MQCD a je předána všem zbývajícím uživatelským procedurám v řetězci uživatelských procedur a všem konverzím sdílejících instanci kanálu. Další informace naleznete v tématu Změna polí MQCD v uživatelské proceduře kanálu .

Také pro programy napsané v jazyce C nesmí být v programu uživatelské procedury kanálu použita funkce knihovny jazyka C, která není reentrantem.

Linux AIX Používáte-li knihovny uživatelských procedur pro více kanálů současně, mohou na některých platformách UNIX and Linux nastat problémy, pokud kód pro dvě různé uživatelské procedury obsahuje identicky pojmenované funkce. Při načtení uživatelské procedury kanálu dynamický zavaděč interpretuje názvy funkcí v knihovně uživatelské procedury na adresy, na které je knihovna zavedena. Pokud dvě výstupní knihovny definují oddělené funkce, které mají shodná jména, může tento proces rozlišení nesprávně interpretovat názvy funkcí jedné knihovny tak, aby používaly funkce jiné. Dojde-li k tomuto problému, určete spojovému programu, že musí exportovat pouze požadované uživatelské procedury a funkce MQStart, protože tyto funkce nejsou ovlivněny. Ostatním funkcím musí být poskytnuta lokální viditelnost, aby je nepoužívaly funkce mimo svou vlastní uživatelskou knihovnu. Další informace naleznete v dokumentaci spojovacího programu.

Všechny uživatelské procedury jsou volány se strukturou parametrů uživatelské procedury kanálu (MQCXP), strukturou definice kanálu (MQCD), připravenou datovou vyrovnávací pamětí, parametrem délky dat a parametrem délky vyrovnávací paměti. Délka vyrovnávací paměti nesmí být překročena:

- V případě uživatelských procedur pro zprávy je nutné povolit odeslání největší požadované zprávy v rámci kanálu plus délku struktury MQXQH.
- Pro uživatelské procedury odeslání a příjmu je největší vyrovnávací paměť, kterou musíte povolit, následující:

  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. Podrobnosti naleznete v hodnotě vrácené v části MaxSegmentLength. Další informace o délce MaxSegmentviz MaxSegmentMaxSegment.

  NetBIOS:
  64 kB

  SPX:
  64 kB

  Poznámka: Uživatelské procedury pro příjem na odesílacích kanálech a uživatelské procedury pro odesílatele na přijímacích kanálech používají 2 kB vyrovnávací paměti pro TCP.
- Pro uživatelské procedury zabezpečení alokuje prostředek distribuovaného řazení do front vyrovnávací paměť o velikosti 4000 bajtů.

Je povoleno, aby výstup vrátil alternativní vyrovnávací paměť spolu s příslušnými parametry. Podrobnosti o volání viz “Programy uživatelské procedury kanálu pro kanály systému zpráv” na stránce 928 .

z/OS *Zápis programů uživatelské procedury kanálu na systému z/OS*

Následující informace vám mohou pomoci při psaní a kompilaci programů uživatelské procedury kanálu pro systém z/OS.

Uživatelské procedury jsou spouštěny jako by byly pomocí z/OS LINK, v:

- Neautorizovaný problémový stav programu
- Režim řízení primárního adresního prostoru
- Nekřížový režim paměti
- Režim registru bez přístupu
- 31bitový režim adresování

Moduly upravené odkazem musí být umístěny v datové sadě určené příkazem CSQXLIB DD procedury adresního prostoru inicializátoru kanálu; názvy zaváděcích modulů jsou určeny jako názvy ukončení v definici kanálu.

Při zápisu uživatelských procedur kanálu pro produkt z/OSplatí následující pravidla:

- Uživatelské procedury musí být napsány v assembleru nebo C; pokud je použit jazyk C, musí odpovídat programovacímu prostředí systémů C pro systémové uživatelské procedury, jak je popsáno v příručce z/OS C/C++ Programming Guide.
- Uživatelské procedury jsou načteny z neautorizovaných knihoven definovaných příkazem CSQXLIB DD. Pokud má CSQXLIB DISP=SHR, lze uživatelské procedury aktualizovat, zatímco je spuštěn inicializátor kanálu. Nová verze se použije při restartování kanálu.
- Uživatelské procedury musí být reentrant a musí být schopné pracovat kdekoli ve virtuálním úložišti.
- Při návratu musí uživatelské procedury resetovat prostředí na prostředí, které je uvedeno v položce.
- Uživatelské procedury musí uvolnit jakékoli získané úložiště, nebo se musí ujistit, že je uvolněno následným vyvoláním uživatelské procedury.

  Pro úložiště, které má zůstat mezi vyvoláními, použijte službu z/OS STORAGE nebo funkci knihovny 4kmalc pro programování systému C.

  Další informace o této funkci naleznete v tématu 4kmalc() -- Přidělit paměť se zarovnanou stránkou.
- Lze použít všechna volání IBM MQ MQI s výjimkou volání MQCMIT nebo CSQBCMT a MQBACK nebo CSQBBAK. Musí být obsaženy za MQCONN (s prázdným názvem správce front). Jsou-li použita tato volání, musí být uživatelská procedura upravena pomocí odkazu se stubem CSQXSTUB.

  Výjimkou z tohoto pravidla je, že uživatelské procedury kanálu zabezpečení mohou vydávat volání MQI pro potvrzení a vrácení. Chcete-li taková volání provést, zadejte místo příkazů MQCMIT nebo CSQBCMT a MQBACK nebo CSQBBAK kód sloves CSQXCMT a CSQXBAK.
- Všechny uživatelské procedury, které používají stub CSQXSTUB z produktu IBM WebSphere MQ 7.0 nebo novější, musí být upraveny pomocí odkazu v zaváděcí knihovně CSQXLIB s formátem PDS-E.
- Uživatelské procedury nesmí používat žádné systémové služby, které způsobují čekání, protože použití systémových služeb by vážně ovlivnilo obsluhu některých nebo všech ostatních kanálů. Mnoho kanálů je obvykle provozováno pod jedním TCB. Pokud v uživatelské proceduře provedete něco, co způsobí čekání, a nepoužijete MQXWAIT, způsobí to, že všechny tyto kanály budou čekat. To, že kanály čekají, nezpůsobuje žádné funkční problémy, ale může mít nepříznivý vliv na výkon. Většina okruhů SVC zahrnuje čekání, takže se jim musíte vyhnout, s výjimkou následujících okruhů SVC:
  - GETMAIN/FREEMAIN/STORAGE
  - LOAD/DELETE

  Obecně tedy vyhněte se SVC, PC a vstupům/výstupům. Místo toho použijte volání MQXWAIT.
- Uživatelské procedury nevydávají ESTAEs nebo SPIEs výjimkou podúloh, které připojují, protože jejich ošetření chyb může kolidovat s ošetřování chyb prováděným produktem IBM MQ. To znamená, že

produkt IBM MQ nemusí být schopen provést zotavení z chyby nebo že uživatelský program neobdrží všechny informace o chybě.

- Volání MQXWAIT (viz MQXWAIT). poskytuje službu čekání, která čeká na I/O a další události; je-li tato služba použita, ukončení nesmí používat zásobník sestavení.

  Pro I/O a další zařízení, která neposkytují neblokující zařízení nebo ECB, aby mohla čekat, musí být oddělená dílčí úloha ATTACHed a její dokončení čekalo na MQXWAIT; kvůli zpracování, které tato technika způsobí, musí být toto zařízení používáno pouze uživatelskou procedurou zabezpečení.

- Volání MQDISC MQI nezpůsobí výskyt implicitního potvrzení v rámci uživatelského programu. Potvrzení procesu kanálu se provádí pouze v případě, že to diktuje protokol kanálu.

Následující ukázky ukončení jsou poskytnuty s produktem IBM MQ for z/OS:

CSQ4BAX0
Tato ukázka je napsána v assembleru a ilustruje použití MQXWAIT.

CSQ4BCX1 a CSQ4BCX2
Tyto ukázky jsou napsány v jazyce C a ilustrují, jak přistupovat k parametrům.

CSQ4BCX3 a CSQ4BAX3
Tyto vzorky jsou napsány v C a assembleru.

Ukázka CSQ4BCX3 (která je předkompilována do knihovny SCSQAUTH LOADLIB, by měla fungovat bez nutnosti změn v samotné uživatelské proceduře. Můžete vytvořit LOADLIB (například s názvem MY.TEST.LOADLIB) a zkopírujte do něj člen SCSQAUTH (CSQ4BCX3).

Chcete-li nastavit uživatelskou proceduru pro zabezpečení připojení klienta, postupujte takto:

1. Zaveďte platný segment OMVS pro ID uživatele, které používá inicializátor kanálu.

   To umožňuje inicializátoru kanálu IBM MQ for z/OS používat protokol TCP/IP s rozhraním soketu z/OS UNIX System Services (z/OS UNIX), aby se usnadnilo zpracování ukončení. Všimněte si, že není nutné definovat segment OMVS pro ID uživatele jakéhokoli připojujícího se klienta.
2. Ujistěte se, že kód ukončení sám běží pouze v programově řízeném prostředí.

   To znamená, že vše načtené do adresního prostoru CHINIT musí být načteno z knihovny řízené programem (což znamená všechny knihovny v knihovně STEPLIB) a všechny knihovny pojmenované v knihovně CSQXLIB a

   ```
   ++hlq++.SCSQANLx
   ++hlq++.SCSQMVR1
   ++hlq++.SCSQAUTH
   ```

   Chcete-li nastavit zaváděcí knihovnu jako řízenou programem, použijte příkaz podobný tomuto příkladu:

   ```
   RALTER PROGRAM * ADDMEM('MY.TEST.LOADLIB'//NOPADCHK)
   ```

   Poté můžete aktivovat nebo aktualizovat programově řízené prostředí zadáním příkazu:

   ```
   SETROPTS WHEN(PROGRAM) REFRESH
   ```
3. Přidejte uživatelskou proceduru LOADLIB do CSQXLIB DD (ve spuštěné proceduře CHINIT) zadáním následujícího příkazu:

   ```
   ALTER CHANNEL(xxxx) CHLTYPE(SVRCONN)SCYEXIT(CSQ4BCX3)
   ```

   Tím se aktivuje uživatelská procedura pro uvedený kanál.
4. Váš externí správce zabezpečení (ESM) uvádí všechny další knihovny, které mají být řízeny programem, ale pamatujte, že žádná z knihoven ESM nebo C nemusí být pod kontrolou programu.

Další informace o nastavení uživatelské procedury pro zabezpečení pomocí ukázky CSQ4BCX3naleznete v tématu IBM MQ for z/OS Kanál připojení serveru .

CSQ4BCX4
Tato ukázka je napsána v jazyce C a demonstruje použití polí **RemoteProduct** a **RemoteVersion** v MQCXP.

Související pojmy

“Zápis programů uživatelské procedury kanálu na systému IBM i” na stránce 935
Následující informace vám mohou pomoci při psaní a kompilaci programů uživatelské procedury kanálu pro systém IBM i.

“Zápis programů uživatelské procedury kanálu na systému AIX, Linux, and Windows” na stránce 936
Následující informace vám mohou pomoci při psaní programů uživatelské procedury kanálu pro systémy AIX, Linux, and Windows .

Související odkazy

IBM MQ for z/OS Kanál připojení serveru

IBM i *Zápis programů uživatelské procedury kanálu na systému IBM i*

Následující informace vám mohou pomoci při psaní a kompilaci programů uživatelské procedury kanálu pro systém IBM i.

Uživatelská procedura je programový objekt napsaný v jazyce ILE C, ILE RPG nebo ILE COBOL. Názvy uživatelských programů a jejich knihovny jsou pojmenovány v definici kanálu.

Při vytváření a kompilaci uživatelského programu dodržujte následující podmínky:

- Program musí být zabezpečen vláknem a vytvořen s kompilátorem ILE C, ILE RPG nebo ILE COBOL. Pro ILE RPG musíte uvést specifikaci řízení THREAD (*SERIALIZE) a pro ILE COBOL musíte uvést SERIALIZE pro volbu THREAD příkazu PROCESS. Programy musí být také svázány s knihovnami IBM MQ s podprocesy: QMQM/LIBMQM_R v případě ILE C a ILE RPG a AMQ0STUB_R v případě ILE COBOL. Další informace o zajištění bezpečnosti podprocesů aplikací v jazycích RPG nebo COBOL naleznete v příslušné programátorské příručce pro daný jazyk.
- Produkt IBM MQ for IBM i vyžaduje, aby uživatelské programy byly povoleny pro podporu teraprostoru. (Teraspace je formou sdílené paměti zavedené v systému OS/400 V4R4.) Pro kompilátory ILE RPG a COBOL jsou všechny programy kompilované na OS/400 V4R4 nebo novější povoleny. Pro C musí být programy kompilovány s volbami TERASPACE (*YES *TSIFC) uvedenými v příkazech CRTCMOD nebo CRTBNDC.
- Uživatelská procedura, která vrací ukazatel na svůj vlastní prostor vyrovnávací paměti, musí zajistit, aby objekt, na který se odkazuje, existoval mimo časové rozpětí programu uživatelské procedury kanálu. Ukazatel nemůže být adresou proměnné v zásobníku programu ani proměnnou v haldě programu. Místo toho musí být ukazatel získán ze systému. Příkladem je uživatelský prostor vytvořený v uživatelské proceduře. Chcete-li zajistit, aby jakákoli datová oblast přidělená programem uživatelské procedury kanálu byla po ukončení programu stále k dispozici pro agenta MCA, musí být uživatelská procedura kanálu spuštěna v aktivační skupině volajícího nebo v pojmenované aktivační skupině. To provedete nastavením parametru ACTGRP v CRTPGM na uživatelsky definovanou hodnotu nebo *CALLER. Pokud je program vytvořen tímto způsobem, může program channel-exit přidělit dynamickou paměť a předat ukazatel na tuto paměť zpět do MCA.

Související pojmy

“Zápis programů uživatelské procedury kanálu na systému AIX, Linux, and Windows” na stránce 936
Následující informace vám mohou pomoci při psaní programů uživatelské procedury kanálu pro systémy AIX, Linux, and Windows .

“Zápis programů uživatelské procedury kanálu na systému z/OS” na stránce 933
Následující informace vám mohou pomoci při psaní a kompilaci programů uživatelské procedury kanálu pro systém z/OS.

ALW *Zápis programů uživatelské procedury kanálu na systému AIX, Linux, and Windows*

Následující informace vám mohou pomoci při psaní programů uživatelské procedury kanálu pro systémy AIX, Linux, and Windows .

Postupujte podle pokynů uvedených v části "Zápis uživatelských procedur a instalovatelných služeb na webu AIX, Linux, and Windows" na stránce 904. V případě potřeby použijte následující specifické informace o uživatelské proceduře kanálu:

Uživatelská procedura musí být napsána v jazyce C a jedná se o knihovnu DLL v systému Windows.

Definujte v uživatelské proceduře fiktivní rutinu MQStart () a zadejte `MQStart` jako vstupní bod v knihovně. Obrázek 106 na stránce 936 ukazuje, jak nastavit položku ve vašem programu:

```
#include <cmqec.h>

void MQStart() {;} /* dummy entry point - for consistency only */
void MQENTRY ChannelExit ( PMQCXP  pChannelExitParms,
                           PMQCD   pChannelDefinition,
                           PMQLONG pDataLength,
                           PMQLONG pAgentBufferLength,
                           PMQVOID pAgentBuffer,
                           PMQLONG pExitBufferLength,
                           PMQPTR  pExitBufferAddr)
{
... Insert code here
}
```

Obrázek 106. Ukázkový zdrojový kód pro uživatelskou proceduru kanálu

Při zápisu uživatelských procedur kanálu pro systém Windows pomocí Visual C + + musíte napsat svůj vlastní soubor DEF . Příklad, jak je zobrazeno v souboru Obrázek 107 na stránce 936. Další informace o psaní programů uživatelské procedury kanálu naleznete v části "Psaní programů uživatelské procedury kanálu" na stránce 931.

```
EXPORTS
ChannelExit
```

Obrázek 107. Ukázkový soubor DEF pro Windows

Související pojmy

"Zápis programů uživatelské procedury kanálu na systému IBM i" na stránce 935
Následující informace vám mohou pomoci při psaní a kompilaci programů uživatelské procedury kanálu pro systém IBM i.

"Zápis programů uživatelské procedury kanálu na systému z/OS" na stránce 933
Následující informace vám mohou pomoci při psaní a kompilaci programů uživatelské procedury kanálu pro systém z/OS.

Uživatelské programy zabezpečení kanálu
Můžete použít uživatelské procedury zabezpečení, abyste ověřili, že partner na druhém konci kanálu je pravý. Toto je známé jako ověření.

Chcete-li určit, že kanál musí používat uživatelskou proceduru pro zabezpečení zprávy, zadejte název uživatelské procedury do pole **SCYEXIT** v definici kanálu.

Poznámka: Ověření lze také dosáhnout pomocí záznamů ověření kanálu. Záznamy ověření kanálu poskytují velkou flexibilitu při prevenci přístupu ke správcům front z určitých uživatelů a kanálů a při mapování vzdálených uživatelů na identifikátory uživatelů IBM MQ . Podporu TLS poskytuje také produkt IBM MQ k ověření uživatelů a k zajištění kontroly 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 potřebujete sofistikovanější (nebo různé) formy zpracování zabezpečení a další typy kontrol a vytváření kontextu zabezpečení, zvažte možnost psaní uživatelských procedur zabezpečení.

Atributy DN subjektu a vydavatele se zobrazují v následujících atributech stavu kanálu:

- SSLPEER (selektor PCF MQCACH_SSL_SHORT_PEER_NAME)
- SSLCERTI (selektor PCF MQCACH_SSL_CERT_ISSUER_NAME)

Tyto hodnoty jsou vráceny příkazy stavu kanálu a také daty předanými do uživatelských procedur zabezpečení kanálu, jak je uvedeno:

- MQCD SSLPeerNamePtr
- MQCXP SSLRemCertIssNamePtr

Uživatelskou proceduru pro zabezpečení zprávy lze zapsat v jazyce C nebo Java.

Uživatelské procedury zabezpečení kanálu jsou volány na následujících místech v cyklu zpracování agenta MCA:

- Při zahájení a ukončení MCA.
- Okamžitě po dokončení počátečního vyjednávání dat při spuštění kanálu. Přijímající nebo serverový konec kanálu může zahájit výměnu zpráv zabezpečení se vzdáleným koncem poskytnutím zprávy, která má být doručena uživatelské proceduře zabezpečení na vzdáleném konci. Mohlo by to také odmítnout. Uživatelský program se znovu spustí, aby zpracoval jakoukoli zprávu zabezpečení přijatou ze vzdáleného konce.
- Okamžitě po dokončení počátečního vyjednávání dat při spuštění kanálu. Odesilatel nebo žadatel na konci kanálu zpracuje zprávu zabezpečení přijatou ze vzdáleného konce nebo zahájí výměnu zabezpečení, když to vzdálený konec nemůže. Uživatelský program se znovu spustí, aby zpracoval všechny následné zprávy zabezpečení, které mohou být přijaty.

Žadatelský kanál není nikdy volán s MQXR_INIT_SEC. Kanál oznámí serveru, že má program uživatelské procedury pro zabezpečení zprávy, a server pak bude mít možnost zahájit uživatelskou proceduru pro zabezpečení zprávy. Pokud žádný nemá, informuje žadatele a do uživatelského programu se vrátí tok s nulovou délkou.

Poznámka: Vyvarujte se odesílání zpráv zabezpečení s nulovou délkou.

Příklady dat vyměňovaných programy ukončení zabezpečení jsou znázorněny na obrázcích Obrázek 108 na stránce 938 až Obrázek 111 na stránce 940. V těchto příkladech je uvedena posloupnost událostí, ke kterým dochází v rámci uživatelské procedury pro zabezpečení zprávy příjemce a v rámci uživatelské procedury pro zabezpečení zprávy odesilatele. Následující řádky na obrázcích představují plynutí času. V některých případech nejsou události u příjemce a odesilatele korelovány, a proto se mohou vyskytnout současně nebo v různých časech. V jiných případech má událost v jednom výstupním programu za následek doplňkovou událost, která se vyskytne později v druhém výstupním programu. Například v souboru Obrázek 108 na stránce 938:

1. Příjemce a odesilatel jsou vyvoláni pomocí MQXR_INIT, ale tato vyvolání nejsou korelována, a proto se mohou vyskytovat ve stejnou dobu nebo v různou dobu.
2. Příjemce je dále vyvolán s MQXR_INIT_SEC, ale vrací MQXCC_OK, který nevyžaduje žádnou doplňkovou událost v uživatelské proceduře odesilatele.
3. Odesilatel je dále vyvolán s MQXR_INIT_SEC. Toto není korelováno s vyvoláním příjemce s MQXR_INIT_SEC. Odesilatel vrátí MQXCC_SEND_SEC_MSG, což způsobí doplňkovou událost v uživatelské proceduře příjemce.
4. Příjemce je poté vyvolán s MQXR_SEC_MSG a vrací MQXCC_SEND_SEC_MSG, což způsobí doplňkovou událost v uživatelské proceduře odesilatele.
5. Odesilatel je poté vyvolán s hodnotou MQXR_SEC_MSG a vrací hodnotu MQXCC_OK, která nevyžaduje žádnou doplňkovou událost v uživatelské proceduře příjemce.

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 108. Výměna iniciovaná odesílatelem 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 109. Výměna iniciovaná odesílatelem bez dohody

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 110. Výměna iniciovaná přijímačem se souhlasem

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 111. Výměna iniciovaná příjemcem bez dohody

Programu uživatelské procedury zabezpečení kanálu je předána vyrovnávací paměť agenta obsahující data zabezpečení, s výjimkou záhlaví přenosu generovaných uživatelskou procedurou pro zabezpečení zprávy. Tato data mohou být libovolnými vhodnými daty, aby mohl každý konec kanálu provést ověření zabezpečení.

Program uživatelské procedury zabezpečení na odesílajícím i přijímajícím konci kanálu zpráv může vrátit jeden ze dvou kódů odezvy pro libovolné volání:

- Výměna zabezpečení skončila bez chyb
- Potlačit kanál a zavřít

Poznámka:

1. Uživatelské procedury zabezpečení kanálu obvykle pracují ve dvojicích. Při definování příslušných kanálů se ujistěte, že jsou pro oba konce kanálu pojmenovány kompatibilní uživatelské programy.
2. IBM i V systému IBM i mohou uživatelské programy zabezpečení, které byly kompilovány s parametrem `Use adopted authority` (USEADPAUT = *YES), převzít oprávnění QMQM nebo QMQMADM. Dávejte pozor, aby uživatelská procedura nevyužívala tuto funkci k tomu, aby představovala bezpečnostní riziko pro váš systém.
3. Na kanálu TLS, na kterém druhý konec kanálu poskytuje certifikát, obdrží uživatelská procedura zabezpečení rozlišující název subjektu tohoto certifikátu v poli MQCD, ke kterému přistupuje SSLPeerNamePtr, a rozlišující název vydavatele v poli MQCXP, ke kterému přistupuje SSLRemCertIssNamePtr. Použití, do kterých lze tento název vložit, jsou:
   - Chcete-li omezit přístup prostřednictvím kanálu TLS.
   - Změna MQCD.MCAUserIdentifier na základě názvu.

Související pojmy

Koncepce TLS (Transport Layer Security)

Související odkazy

Záznamy ověření kanálu

Zápis uživatelské procedury zabezpečení

Uživatelskou proceduru pro zabezpečení zprávy můžete napsat pomocí kódu kostry uživatelské procedury pro zabezpečení zprávy.

Obrázek 112 na stránce 941 ukazuje, jak napsat uživatelskou proceduru pro zabezpečení zprávy.

```
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 112. Kód kostry uživatelské procedury zabezpečení

Standardní IBM MQ vstupní bod MQStart musí existovat, ale není vyžadován pro provedení žádné funkce. Název funkce (v tomto příkladuEntryPoint) lze změnit, ale funkci je třeba exportovat, když je knihovna kompilována a propojena. Stejně jako v předchozím příkladu musí být ukazatele pChannelExitParms přetypovány na PMQCXP a definice pChannelmusí být přetypován na PMQCD. Obecné informace o volání uživatelských procedur kanálu a použití parametrů viz MQ_CHANNEL_EXIT. Tyto parametry se používají v uživatelské proceduře zabezpečení následujícím způsobem:

PMQVOID pChannelExitParms
Vstup a výstup

Ukazatel na strukturu MQCXP-pro přístup k polím je přetypován na PMQCXP. Tato struktura se používá ke komunikaci mezi uživatelskou procedurou a agentem MCA. Následující pole v MQCXP mají zvláštní význam pro uživatelské procedury zabezpečení:

ExitReason
Informuje uživatelskou proceduru zabezpečení o aktuálním stavu v rámci výměny zabezpečení a používá se při rozhodování o tom, jaká akce má být provedena.

ExitResponse
Reakce na MCA, která diktuje další fázi bezpečnostní výměny.

ExitResponse2
Další řídicí příznaky, které řídí, jak agent MCA interpretuje odezvu uživatelské procedury zabezpečení.

Oblast ExitUser
16 bajtů (maximum) úložiště, které může uživatelská procedura zabezpečení použít k udržování stavu mezi voláními.

ExitData
Obsahuje data uvedená v poli SCYDATA definice kanálu (32 bajtů vyplněných vpravo mezerami).

Definice PMQVOID pChannel
Vstup a výstup

Ukazatel na strukturu MQCD-přetypování na PMQCD pro přístup k polím. Tento parametr obsahuje definici kanálu. Následující pole v MQCD mají zvláštní význam pro uživatelské procedury zabezpečení:

ChannelName
Název kanálu (20 bajtů napravo s mezerami).

ChannelType
Kód definující typ kanálu.

Identifikátor uživatele MCA
Tato skupina tří polí je inicializována na hodnotu pole MCAUSER uvedenou v definici kanálu. Pro řízení přístupu se použije jakýkoli identifikátor uživatele uvedený v polích Uživatelská procedura zabezpečení (nelze použít pro kanály SDR, SVR, CLNTCONN nebo CLUSSDR).

MCAUserIdentifier
Prvních 12 bajtů identifikátoru vyplněných vpravo mezerami.

LongMCAUserIdPtr
Ukazatel na vyrovnávací paměť obsahující identifikátor celé délky (není zaručeno ukončení hodnoty null) má přednost před parametrem MCAUserIdentifier.

LongMCAUserIdLength
Délka řetězce, na který odkazuje LongMCAUserIdPtr -musí být nastavena, pokud je nastaveno LongMCAUserIdPtr .

Identifikátor vzdáleného uživatele
Platí pouze pro dvojice kanálů CLNTCONN/SVRCONN. Není-li definována žádná uživatelská procedura zabezpečení CLNTCONN, jsou tato tři pole inicializována klientským agentem MCA, takže mohou obsahovat identifikátor uživatele z prostředí klienta, který může být použit uživatelskou procedurou zabezpečení SVRCONN pro ověření a při zadávání identifikátoru uživatele MCA. Je-li definována uživatelská procedura zabezpečení CLNTCONN, nejsou tato pole inicializována a lze je 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 vyplněných vpravo mezerami.

LongRemoteUserIdPtr
Ukazatel na vyrovnávací paměť obsahující identifikátor celé délky (není zaručeno ukončení hodnoty null) má přednost před identifikátorem RemoteUser.

LongRemoteUserIdDélka
Délka řetězce, na který odkazuje LongRemoteUserIdPtr-musí být nastavena, pokud je nastaveno LongRemoteUserIdPtr.

Délka pDataparametru PMQLONG
Vstup a výstup

Ukazatel na MQLONG. Obsahuje délku jakékoli uživatelské procedury zabezpečení obsažené v AgentBuffer při vyvolání uživatelské procedury zabezpečení. Musí být nastaveno uživatelskou procedurou pro zabezpečení zprávy na délku jakékoli zprávy odesílané v AgentBuffer nebo ExitBuffer.

PMQLONG pAgentBufferLength
Vstup

Ukazatel na MQLONG. Délka dat obsažených ve AgentBuffer při vyvolání uživatelské procedury zabezpečení.

Vyrovnávací paměť PMQVOID pAgent
Vstup a výstup

Při vyvolání uživatelské procedury zabezpečení odkazuje na jakoukoli zprávu odeslanou z uživatelské procedury partnera. Má-li volba ExitResponse2 ve struktuře MQCXP nastaven příznak MQXR2_USE_AGENT_BUFFER (výchozí), musí uživatelská procedura zabezpečení nastavit tento parametr tak, aby ukazoval na odesílaná data zprávy.

PMQLONG pExitBufferLength
Vstup a výstup

Ukazatel na MQLONG. Tento parametr je inicializován na hodnotu 0 při prvním vyvolání uživatelské procedury zabezpečení a vrácená hodnota je udržována mezi voláními uživatelské procedury zabezpečení během výměny zabezpečení.

PMQPTR pExitBufferAddr
Vstup a výstup

Tento parametr je inicializován na nulový ukazatel při prvním vyvolání uživatelské procedury zabezpečení a vrácená hodnota je udržována mezi voláními uživatelské procedury zabezpečení během výměny zabezpečení. Je-li ve struktuře ExitResponse2 ve struktuře MQCXP nastaven příznak MQXR2_USE_EXIT_BUFFER , musí uživatelská procedura zabezpečení nastavit tento parametr tak, aby ukazoval na odesílaná data zprávy.

Rozdíly v chování mezi bezpečnostními ukončeními definovanými ve dvojicích kanálů CLNTCONN/SVRCONN a ostatními dvojicemi kanálů
Uživatelské procedury zabezpečení lze definovat pro všechny typy kanálů. Avšak chování uživatelských procedur zabezpečení definovaných ve dvojicích kanálů CLNTCONN/SVRCONN se mírně liší od uživatelských procedur zabezpečení definovaných v jiných dvojicích kanálů.

Uživatelská procedura zabezpečení na kanálu CLNTCONN může nastavit identifikátor vzdáleného uživatele v definici kanálu pro zpracování uživatelskou procedurou SVRCONN partnera nebo pro autorizaci OAM, pokud není definována žádná uživatelská procedura zabezpečení SVRCONN a není nastaveno pole MCAUSER SVRCONN.

Pokud není 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) klientským agentem MCA.

Výměna zabezpečení mezi uživatelskými procedurou zabezpečení definovanými ve dvojici kanálů CLNTCONN a SVRCONN se úspěšně dokončí, když uživatelská procedura zabezpečení SVRCONN vrátí odezvu ExitResponse MQXCC_OK. Výměna zabezpečení mezi ostatními dvojicemi kanálů se úspěšně dokončí, když uživatelská procedura zabezpečení, která zahájila výměnu, vrátí hodnotu ExitResponse MQXCC_OK.

Avšak kód MQXCC_SEND_AND_REQUEST_SEC_MSG ExitResponse lze použít k vynucení pokračování výměny zabezpečení: Pokud je ExitResponse MQXCC_SEND_AND_REQUEST_SECEST_MSG vrácen uživatelskou procedurou zabezpečení CLNTCONN nebo SVRCONN, pak musí uživatelská procedura

partnera odpovědět odesláním zprávy zabezpečení (ne MQXCC_OK nebo null), nebo se kanál ukončí. V případě uživatelských procedur zabezpečení definovaných na jiných typech kanálu má volba ExitResponse MQXCC_OK jako odpověď na MQXCC_SEND_AND_REQUEST_SEC_MSG z partnerské uživatelské procedury zabezpečení za následek pokračování výměny zabezpečení, jako by byla vrácena nulová odezva, a nikoli ukončení kanálu.

Uživatelská procedura zabezpečení rozhraní SSPI
Produkt IBM MQ for Windows poskytuje uživatelskou proceduru zabezpečení, která poskytuje ověření pro kanály IBM MQ pomocí rozhraní SSPI (Security Services Programming Interface). Rozhraní SSPI poskytuje integrované prostředky zabezpečení produktu Windows.

Tato uživatelská procedura zabezpečení je určena pro klienta IBM MQ i pro server IBM MQ .

Balíky zabezpečení jsou načteny buď z knihovny security.dll , nebo z knihovny secur32.dll. Tyto knihovny DLL jsou dodávány s vaším operačním systémem.

V systému Windowsse poskytuje jednosměrné ověření pomocí ověřovacích služeb NTLM. V systému Windows 2000se poskytuje obousměrné ověření pomocí ověřovacích služeb Kerberos .

Uživatelský program zabezpečení je dodáván ve formátu zdroje a objektu. Můžete použít objektový kód tak, jak je, nebo můžete použít zdrojový kód jako výchozí bod pro vytvoření vlastních uživatelských programů. Další informace o použití objektu nebo zdrojového kódu uživatelské procedury pro zabezpečení rozhraní SSPI naleznete v tématu “Použití uživatelské procedury zabezpečení rozhraní SSPI v systému Windows” na stránce 1096

Uživatelské programy pro odesílání a příjem kanálu
Uživatelské procedury odeslání a přijetí můžete použít k provedení úloh, jako je komprese a dekomprese dat. Můžete určit seznam programů uživatelských procedur pro odeslání a příjem, které mají být spuštěny v posloupnosti.

Uživatelské programy pro odesílání a příjem kanálů jsou volány na následujících místech v cyklu zpracování MCA:

- Uživatelské programy odeslání a přijetí jsou volány pro inicializaci při inicializaci MCA a pro ukončení při ukončení MCA.
- Uživatelský program odeslání je vyvolán na jednom nebo druhém konci kanálu v závislosti na konci, na kterém je odeslán přenos pro jeden přenos zprávy, bezprostředně před odesláním přenosu přes linku. Poznámka 4 vysvětluje, proč jsou východy k dispozici v obou směrech, i když kanály zpráv odesílají zprávy pouze v jednom směru.
- Uživatelský program příjmu je vyvolán na jednom nebo druhém konci kanálu v závislosti na konci, na kterém je přijat přenos pro jeden přenos zprávy, bezprostředně po přenosu z linky. Poznámka 4 vysvětluje, proč jsou východy k dispozici v obou směrech, i když kanály zpráv odesílají zprávy pouze v jednom směru.

Pro jeden přenos zpráv může existovat mnoho přenosů a může existovat mnoho iterací uživatelských procedur pro odeslání a příjem, než zpráva dosáhne uživatelské procedury pro zprávu na přijímacím konci.

Uživatelské programy kanálu pro odesílání a příjem jsou předány vyrovnávací paměti agenta obsahující data přenosu tak, jak byla odeslána nebo přijata z komunikačního propojení. Pro uživatelské programy odesílání je prvních 8 bajtů vyrovnávací paměti vyhrazeno pro použití agentem MCA a nesmí být změněno. Pokud program vrátí jinou vyrovnávací paměť, pak prvních 8 bajtů musí existovat v nové vyrovnávací paměti. Formát dat prezentovaných uživatelským programům není definován.

Programy uživatelských procedur pro odeslání a příjem musí vrátit dobrý kód odezvy. Jakákoli jiná odpověď způsobí abnormální ukončení MCA (nestandardní ukončení).

Poznámka: Nevyvolávejte volání MQGET, MQPUT ani MQPUT1 v rámci synchronizačního bodu z uživatelské procedury pro odeslání nebo příjem.

Poznámka:

1. Uživatelské procedury odeslání a příjmu obvykle pracují ve dvojicích. Například uživatelská procedura odeslání může komprimovat data a uživatelská procedura přijetí je dekomprimovat, nebo uživatelská

procedura odeslání může data zašifrovat a uživatelská procedura přijetí je dešifrovat. Při definování příslušných kanálů se ujistěte, že jsou pro oba konce kanálu pojmenovány kompatibilní uživatelské programy.

2. Je-li pro kanál zapnuta komprese, jsou uživatelské procedury předávány komprimovaná data.
3. Uživatelské procedury odeslání a příjmu kanálu mohou být volány pro jiné segmenty zpráv než pro data aplikace, například pro stavové zprávy. Nejsou volány během dialogového okna spuštění ani fáze kontroly zabezpečení.
4. Ačkoli kanály zpráv odesílají zprávy pouze v jednom směru, data řízení kanálu, jako jsou například srdeční tepny a konec dávkového zpracování, toky v obou směrech a tyto uživatelské procedury jsou také k dispozici v obou směrech. Některé z počátečních datových toků spuštění kanálu jsou však osvobozeny od zpracování kterýmkoli z uživatelských procedur.
5. Existují okolnosti, za kterých mohou být uživatelské procedury odeslání a přijetí vyvolány mimo pořadí; například, pokud spouštíte řadu uživatelských programů nebo spouštíte také uživatelské procedury zabezpečení. Poté, když je nejprve volána uživatelská procedura pro příjem, aby zpracovala data, může přijmout data, která neprošla odpovídající uživatelskou procedurou pro odesílání. Pokud by uživatelská procedura příjmu právě provedla operaci, například dekompresi, aniž by nejprve zkontrolovala, zda byla požadována, výsledky by byly neočekávané.

   Musíte kódovat své uživatelské procedury pro odeslání a příjem takovým způsobem, aby uživatelská procedura pro příjem mohla zkontrolovat, že data, která přijímá, byla zpracována příslušnou uživatelskou procedurou pro odesílání. Doporučený způsob, jak to udělat, je kódovat své uživatelské programy tak, aby:

   - Uživatelská procedura odeslání nastaví hodnotu devátého bajtu dat na 0 a posune všechna data o 1 bajt, než provede operaci. (První 8 bajtů je vyhrazeno pro použití agentem MCA.)
   - Pokud uživatelská procedura příjmu přijme data, která mají 0 v bajtu 9, ví, že data pocházejí z uživatelské procedury odeslání. Odebere hodnotu 0, provede komplementární operaci a posune výsledná data zpět o 1 bajt.
   - Pokud uživatelská procedura pro příjem přijme data, která mají v bajtu 9 jinou hodnotu než 0, předpokládá, že uživatelská procedura pro odesílání nebyla spuštěna, a odešle data zpět volajícímu beze změny.

   Při použití uživatelských procedur zabezpečení, pokud je kanál ukončen uživatelskou procedurou pro zabezpečení zprávy, je možné, že uživatelská procedura pro odesílání bude volána bez příslušné uživatelské procedury pro příjem. Jedním ze způsobů, jak tomuto problému zabránit, je kódovat uživatelskou proceduru zabezpečení tak, aby nastavila příznak, například v souboru MQCD.SecurityUserData nebo MQCD.SendUserData, když se uživatelská procedura rozhodne ukončit kanál. Pak uživatelská procedura odeslání musí toto pole zkontrolovat a zpracovat data pouze v případě, že příznak není nastaven. Tato kontrola brání uživatelské proceduře pro odesílání v tom, aby zbytečně měnila data, a tak zabraňuje jakýmkoli chybám konverze, které by se mohly vyskytnout, pokud by uživatelská procedura zabezpečení přijala změněná data.

Uživatelské programy odeslání kanálu-rezervace prostoru
Uživatelské procedury odeslání a přijetí můžete použít k transformaci dat před přenosem. Uživatelské procedury odeslání kanálu mohou přidávat vlastní data o transformaci tím, že rezervují prostor v přenosové vyrovnávací paměti.

Tato data jsou zpracována uživatelským programem pro příjem a poté odebrána z vyrovnávací paměti. Můžete například chtít šifrovat data a přidat bezpečnostní klíč pro dešifrování.

Jak si rezervovat prostor a používat jej

Při volání uživatelského programu pro odesílání pro inicializaci nastavte pole *ExitSpace* MQXCP na počet bajtů, které mají být vyhrazeny. Podrobnosti viz MQCXP . Parametr *ExitSpace* lze nastavit pouze během inicializace, tj. když má parametr *ExitReason* hodnotu MQXR_INIT. Je-li uživatelská procedura pro odesílání vyvolána bezprostředně před přenosem s parametrem *ExitReason* nastaveným na hodnotu

MQXR_XMIT, jsou *ExitSpace* bajty vyhrazeny ve vyrovnávací paměti pro přenos. *ExitSpace* není na systému z/OSpodporován.

Uživatelská procedura odeslání nemusí používat celý vyhrazený prostor. Může používat méně než *ExitSpace* bajtů, nebo není-li přenosová vyrovnávací paměť plná, může uživatelská procedura použít více než vyhrazené množství. Při nastavení hodnoty *ExitSpace*musíte ponechat alespoň 1 kB pro data zprávy ve vyrovnávací paměti přenosu. Výkon kanálu může být ovlivněn, pokud je vyhrazený prostor použit pro velké množství dat.

Přenosová vyrovnávací paměť je obvykle 32KB dlouhá. Pokud však kanál používá protokol TLS, velikost přenosové vyrovnávací paměti se sníží na 15 352 bajtů, aby se vešla do maximální délky záznamu definované standardem RFC 6101 a související rodinou standardů TLS. Dalších 1024 bajtů je vyhrazeno pro použití produktem IBM MQ, takže maximální prostor přenosové vyrovnávací paměti použitelný pro uživatelské procedury odeslání je 14 328 bajtů.

Co se stane na přijímacím konci kanálu

Uživatelské programy pro příjem kanálů musí být nastaveny tak, aby byly kompatibilní s odpovídajícími uživatelskými procedurami pro odesílání. Uživatelské procedury pro příjem musí znát počet bajtů ve vyhrazeném prostoru a musí odebrat data v tomto prostoru.

Vícenásobné uživatelské procedury odeslání

Můžete určit seznam programů uživatelských procedur pro odeslání a příjem, které mají být spuštěny v posloupnosti. Produkt IBM MQ udržuje celkový prostor vyhrazený pro všechny uživatelské procedury odeslání. Tento celkový prostor musí zůstat alespoň 1 kB pro data zprávy v přenosové vyrovnávací paměti.

Následující příklad ukazuje, jak je prostor přidělen pro tři uživatelské procedury odeslání, volané postupně:

1. Při volání pro inicializaci:
   - Odeslání exit A rezervuje 1 KB.
   - Odeslání exit B rezervuje 2 KB.
   - Odeslání ukončení C rezervuje 3 KB.
2. Maximální velikost přenosu je 32 kB a uživatelská data jsou dlouhá 5 kB.
3. Ukončení A je voláno s 5 kB dat; k dispozici je až 27 kB, protože 5 kB je vyhrazeno pro ukončení B a C. Exit A přidá 1 kB, což je množství, které rezervuje.
4. Ukončení B je voláno s 6 kB dat; k dispozici je až 29 kB, protože 3 kB je vyhrazeno pro ukončení C. Ukončení B přidá 1 kB, což je méně než 2 kB, které rezervuje.
5. Ukončení C je voláno se 7 KB dat; k dispozici je až 32 KB. Ukončení C přidá hodnotu 10K, což je více než 3 kB, které rezervuje. Toto množství je platné, protože celkové množství dat, 17 kB, je menší než maximum 32 kB.

Maximální velikost přenosové vyrovnávací paměti pro kanál používající protokol TLS je 15 352 bajtů, nikoli 32KB. Důvodem je, že základní segmenty přenosu zabezpečeného soketu jsou omezeny na 16KB a část prostoru je požadována pro režijní náklady záznamu TLS. Dalších 1024 bajtů je vyhrazeno pro použití produktem IBM MQ, takže maximální prostor přenosové vyrovnávací paměti použitelný pro uživatelské procedury odeslání je 14 328 bajtů.

Uživatelské programy pro zprávy kanálu
Uživatelskou proceduru pro zprávy kanálu můžete použít k provedení úloh, jako je šifrování odkazu, ověření platnosti nebo nahrazení příchozích ID uživatelů, konverze dat zpráv, žurnálování a zpracování referenčních zpráv. Můžete uvést seznam programů uživatelských procedur pro zprávy, které se mají spustit po sobě.

Programy uživatelské procedury pro zprávy kanálu jsou volány na následujících místech v cyklu zpracování agenta MCA:

- Při zahájení a ukončení MCA

- Okamžitě poté, co odesílající agent MCA vydal volání MQGET
- Před tím, než přijímající agent MCA vydá volání MQPUT

Uživatelské proceduře pro zprávy je předána vyrovnávací paměť agenta obsahující záhlaví přenosové fronty MQXQH a text zprávy aplikace načtený z fronty. Formát MQXQH je uveden v MQXQH-záhlaví přenosové fronty.

Používáte-li referenční zprávy (tj. zprávy obsahující pouze záhlaví, které odkazuje na jiný objekt, který má být odeslán), uživatelská procedura pro zprávy rozpozná záhlaví MQRMH. Identifikuje objekt, načte jej jakýmkoli způsobem, který je vhodný, připojí jej k záhlaví a předá jej do MCA pro přenos do přijímajícího MCA. V přijímající oblasti MCA jiná uživatelská procedura zprávy rozpozná, že tato zpráva je referenční zpráva, extrahuje objekt a předá záhlaví do cílové fronty. Další informace o referenčních zprávách a některých ukázkových uživatelských procedur pro zprávy, které je obsluhují, naleznete v části “Referenční zprávy” na stránce 769 a “Spuštění ukázek referenční zprávy” na stránce 1070 .

Uživatelské procedury zpráv mohou vrátit následující odpovědi:

- Odešlete zprávu (uživatelská procedura GET). Zpráva mohla být změněna uživatelskou procedurou. (To vrací MQXCC_OK.)
- Vložte zprávu do fronty (uživatelská procedura PUT). Zpráva mohla být změněna uživatelskou procedurou. (To vrací MQXCC_OK.)
- Nezpracovávejte zprávu. Zpráva je umístěna do fronty nedoručených zpráv (nedoručená fronta zpráv) agentem MCA.
- Zavřete kanál.
- Chybný návratový kód, který způsobí nestandardní ukončení agenta MCA.

Poznámka:

1. Uživatelské procedury zpráv jsou volány jednou pro každou přenesenou kompletní zprávu, i když je zpráva rozdělena na části.
2. Linux AIX Zadáte-li uživatelskou proceduru pro zprávy v systému AIX nebo Linux, nebude automatický převod ID uživatelů na malá písmena (zde) fungovat.
3. Uživatelská procedura se spustí ve stejném podprocesu jako samotný agent MCA. Také běží ve stejné pracovní jednotce (UOW) jako MCA, protože používá stejný manipulátor připojení. Takže všechna volání provedená v synchronizačním bodu jsou potvrzena nebo zálohována kanálem na konci dávky. Například jeden program uživatelské procedury pro zprávy kanálu může odesílat oznamovací zprávy do jiného a tyto zprávy jsou potvrzeny do fronty pouze v případě, že je dávka obsahující původní zprávu potvrzena.

   Proto můžete spustit volání MQI synchronizačního bodu z programu uživatelské procedury pro zprávy kanálu.

Převod zpráv mimo uživatelskou proceduru zprávy
Před voláním uživatelské procedury pro zprávu provádí přijímající agent MCA některé konverze zprávy. Toto téma popisuje algoritmy používané k provádění převodů.

Která záhlaví jsou zpracována

Rutina převodu se spustí v MCA příjemce před zavoláním uživatelské procedury pro zprávu. Rutina převodu začíná záhlavím MQXQH na začátku zprávy. Rutina převodu poté zpracuje prostřednictvím zřetězených záhlaví, která následují za MQXQH, a v případě potřeby provede převod. Zřetězená záhlaví mohou přesahovat posunutí obsažené v parametru HeaderLength dat MQCXP předávaných uživatelské proceduře pro zprávy příjemce. Následující záhlaví jsou převedena na místě:

- MQXQH (název formátu " `MQXMIT` ")
- MQMD (toto záhlaví je součástí MQXQH a nemá název formátu)
- MQMDE (název formátu " `MQHMDE` ")

- MQDH (název formátu " MQHDIST ")
- MQWIH (název formátu " MQHWIH ")

Následující záhlaví nejsou převedena, ale jsou překročena, protože agent MCA pokračuje ve zpracování zřetězených záhlaví:

- MQDLH (název formátu " MQDEAD ")
- libovolná záhlaví s názvy formátů začínajícími třemi znaky 'MQH' (například " MQHRF ") které nejsou jinak zmíněny

Způsob zpracování záhlaví

Parametr Formát každého záhlaví IBM MQ je načten agentem MCA. Parametr Formát je 8 bajtů v záhlaví, což je 8 jednobajtových znaků obsahujících název.

Agent MCA poté interpretuje data za každým záhlavím jako pojmenovaný typ. Pokud je formát název typu záhlaví vhodný pro převod dat IBM MQ , převede se. Pokud se jedná o jiný název označující jiná data nežMQ (například MQFMT_NONE nebo MQFMT_STRING), pak MCA zastaví zpracování záhlaví.

Co je MQCXP HeaderLength?

Parametr HeaderLength v datech MQCXP dodaných uživatelské proceduře pro zprávu je celková délka záhlaví MQXQH (včetně MQMD), MQMDE a MQDH na začátku zprávy. Tato záhlaví jsou zřetězena pomocí názvů a délek 'Formát'.

MQWIH

Zřetězená záhlaví mohou přesahovat hodnotu HeaderLength do oblasti uživatelských dat. Záhlaví MQWIH, je-li přítomno, je jedním z těch záhlaví, která se zobrazují za záhlavím HeaderLength.

Pokud je v zřetězených záhlavích záhlaví MQWIH, je převedeno na místě před zavoláním uživatelské procedury pro zprávu příjemce.

Uživatelský program opakování zpráv kanálu
Uživatelská procedura opakování zpráv kanálu je volána, když je pokus o otevření cílové fronty neúspěšný. Uživatelskou proceduru můžete použít k určení, za jakých okolností se má opakovat, kolikrát se má opakovat a jak často.

Tato uživatelská procedura je také volána na přijímacím konci kanálu při zahájení a ukončení MCA.

Uživatelské proceduře pro opakování zpráv kanálu je předána vyrovnávací paměť agenta obsahující záhlaví přenosové fronty MQXQH a text zprávy aplikace načtený z fronty. Formát MQXQH je uveden v části Přehled pro MQXQH.

Uživatelská procedura je vyvolána pro všechny kódy příčiny; uživatelská procedura určuje, pro které kódy příčiny chce agenta MCA opakovat, kolikrát a v jakých intervalech. (Hodnota počtu opakování zpráv nastavená při definování kanálu je předána uživatelské proceduře v MQCD, uživatelská procedura však může tuto hodnotu ignorovat.)

Pole Počet MsgRetryv MQCXP je při každém vyvolání uživatelské procedury zvýšeno o hodnotu MCA a uživatelská procedura vrátí buď hodnotu MQXCC_OK s dobou čekání obsaženou v poli Interval MsgRetryMQCXP, nebo hodnotu MQXCC_SUPPRESS_FUNCTION. Opakované pokusy budou pokračovat nekonečně dlouho, dokud uživatelská procedura nevrátí hodnotu MQXCC_SUPPRESS_FUNCTION v poli ExitResponse v prostředí MQCXP. Informace o akci, kterou agent MCA provede pro tyto kódy dokončení, naleznete v tématu MQCXP .

Pokud jsou všechny opakované pokusy neúspěšné, zpráva se zapíše do fronty nedoručených zpráv. Pokud není k dispozici žádná fronta nedoručených zpráv, kanál se zastaví.

Pokud nedefinujete uživatelskou proceduru pro opakování zpráv pro kanál a dojde k selhání, které bude pravděpodobně dočasné, například MQRC_Q_FULL, agent MCA použije počet opakování zpráv a intervaly

opakování zpráv nastavené při definování kanálu. Pokud je selhání trvalejší a nedefinovali jste uživatelský program, který by jej zpracoval, zpráva se zapíše do fronty nedoručených zpráv.

Uživatelský program automatické definice kanálu
Uživatelskou proceduru automatické definice kanálu lze použít, když je přijat požadavek na spuštění kanálu připojení serveru nebo příjemce, ale pro tento kanál neexistuje žádná definice (ne pro IBM MQ for z/OS). Lze jej také volat na všech platformách pro kanály odesílatele klastru a přijímací kanály klastru, aby bylo možné upravit definici pro instanci kanálu.

Uživatelskou proceduru automatické definice kanálu lze volat na všech platformách kromě platformy z/OS , když je přijat požadavek na spuštění kanálu připojení serveru nebo příjemce, ale neexistuje žádná definice kanálu. Můžete jej použít k úpravě dodané výchozí definice pro automaticky definovaný příjemce nebo kanál připojení serveru SYSTEM.AUTO.RECEIVERnebo SYSTEM.AUTO.SVRCON. Popis způsobu automatického vytváření definic kanálů naleznete v tématu Příprava kanálů .

Uživatelská procedura automatické definice kanálu může být také volána, když je přijat požadavek na spuštění odesílacího kanálu klastru. Lze ji volat pro odesílací a přijímací kanály klastru, aby bylo možné upravit definici pro tuto instanci kanálu. V tomto případě se ukončení vztahuje také na IBM MQ for z/OS. Běžné použití uživatelské procedury automatické definice kanálu je změnit názvy uživatelských procedur pro zprávy (MSGEXIT, RCVEXIT, SCYEXIT a SENDEXIT), protože názvy uživatelských procedur mají různé formáty na různých platformách. Není-li uvedena žádná uživatelská procedura automatické definice kanálu, výchozí chování v systému z/OS je prozkoumat název distribuované uživatelské procedury ve tvaru *[path]/libraryname(function)* a mít až osm znaků funkce, je-li přítomna, nebo názvu knihovny. V systému z/OSmusí uživatelský program automatické definice kanálu pozměnit pole adresovaná pomocí polí MsgExitPtr, MsgUserDataPtr, SendExitPtr, SendUserDataPtr, ReceiveExitPtr a ReceiveUserDataPtr, spíše než MsgExit, MsgUserData, SendExit, SendUserData, ReceiveExit a samotná datová pole ReceiveUser.

Další informace naleznete v tématu Práce s automaticky definovanými kanály.

Stejně jako u ostatních uživatelských procedur kanálu je seznam parametrů:

```
MQ_CHANNEL_AUTO_DEF_EXIT (ChannelExitParms, ChannelDefinition)
```

`ChannelExitParms` jsou popsány v části MQCXP. `ChannelDefinition` je popsáno v tématu MQCD.

MQCD obsahuje hodnoty, které jsou použity ve výchozí definici kanálu, pokud nejsou změněny uživatelskou procedurou. Uživatelská procedura může upravit pouze podmnožinu polí; viz MQ_CHANNEL_AUTO_DEF_EXIT. Avšak pokus o změnu jiných polí nezpůsobí chybu.

Uživatelská procedura automatické definice kanálu vrací odezvu MQXCC_OK nebo MQXCC_SUPPRESS_FUNCTION. Pokud není vrácena žádná z těchto odpovědí, agent MCA pokračuje ve zpracování, jako by byla vrácena hodnota MQXCC_SUPPRESS_FUNCTION. To znamená, že automatická definice je opuštěna, není vytvořena žádná nová definice kanálu a kanál nelze spustit.

ALW *Kompilace programů uživatelské procedury kanálu v systémech AIX, Linux, and Windows*

Následující příklady vám pomohou při kompilaci programů uživatelské procedury kanálu pro systémy AIX, Linux, and Windows .

Windows

Windows

Příkaz kompilátoru a spojovacího programu pro programy uživatelské procedury kanálu v systému 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 AIX and Linux

Linux AIX

V těchto příkladech `exit` je název knihovny a `ChannelExit` je název funkce. V systému AIX se soubor exportu nazývá `exit.exp`. Tyto názvy používá definice kanálu k odkazování na uživatelský program s použitím formátu popsaného v tématu Definice kanálu MQCD. Viz také parametr MSGEXIT příkazu DEFINE CHANNEL .

AIX Ukázkové příkazy kompilátoru a spojovacího programu pro uživatelské procedury kanálu v systému AIX:

```
$ xlc_r -q64 -e MQStart -bE:exit.exp -bM:SRE -o /var/mqm/exits64/exit
exit.c -I/usr/mqm/inc
```

Linux Ukázkové příkazy kompilátoru a spojovacího programu pro uživatelské procedury kanálu v systému Linux , kde je správce front 32bitový:

```
$ gcc -shared -fPIC -o /var/mqm/exits/exit exit.c -I/opt/mqm/inc
```

Linux Ukázkové příkazy kompilátoru a spojovacího programu pro uživatelské procedury kanálu v systému Linux , kde je správce front 64bitový:

```
$ gcc -m64 -shared -fPIC -o /var/mqm/exits64/exit exit.c -I/opt/mqm/inc
```

Na klientovi lze použít 32bitovou nebo 64bitovou uživatelskou proceduru. Tato uživatelská procedura musí být propojena s mqic_r.

AIX V systému AIXmusí být exportovány všechny funkce, které jsou volány produktem IBM MQ . Ukázkový soubor exportu pro tento soubor make:

```
#
!channelExit
MQStart
```

Konfigurace uživatelských procedur kanálu

Chcete-li volat uživatelskou proceduru kanálu, musíte ji pojmenovat v definici kanálu.

Uživatelské procedury kanálu musí být pojmenovány v definici kanálu. Toto pojmenování můžete provést při prvním definování kanálů, nebo můžete informace přidat později pomocí například příkazu MQSC ALTER CHANNEL. Můžete také zadat názvy uživatelských procedur kanálu v datové struktuře kanálu MQCD. Formát názvu uživatelské procedury závisí na vaší platformě IBM MQ . Informace naleznete v části MQCD nebo Příkazy MQSC .

Pokud definice kanálu neobsahuje název programu uživatelské procedury, uživatelská procedura se nevolá.

Uživatelská procedura automatické definice kanálu je vlastností správce front, nikoli jednotlivého kanálu. Aby byla 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 převodu dat

Tato kolekce témat obsahuje informace o tom, jak zapisovat uživatelské procedury pro převod dat.

Poznámka: Není podporováno v produktu MQSeries pro VSE/ESA.

Když provedete MQPUT, aplikace vytvoří deskriptor zprávy (MQMD) zprávy. Vzhledem k tomu, že produkt IBM MQ musí být schopen porozumět obsahu deskriptoru MQMD bez ohledu na platformu, na které byl vytvořen, systém jej automaticky převede.

Data aplikace však nejsou automaticky převedena. Pokud se znaková data vyměňují mezi platformami, kde se pole `CodedCharSetId` a `Encoding` liší, například mezi ASCII a EBCDIC, musí aplikace zajistit převod zprávy. Převod dat aplikace může provádět sám správce front nebo program uživatelské procedury označovaný jako *uživatelská procedura pro převod dat*. Pokud jsou data aplikace v jednom z vestavěných formátů (například MQFMT_STRING), může správce front provést převod dat sám pomocí jedné z vestavěných převodních rutin. Toto téma obsahuje informace o prostředku uživatelské procedury pro převod dat, který produkt IBM MQ poskytuje v případě, že data aplikace nejsou ve vestavěném formátu.

Řízení lze předat uživatelské proceduře pro převod dat během volání MQGET. Tím se zabrání převodu mezi různými platformami před dosažením konečného cíle. Je-li však konečným cílem platforma, která nepodporuje převod dat v příkazu MQGET, je třeba zadat hodnotu CONVERT (YES) v kanálu odesilatele, který odesílá data do příslušného konečného místa určení. To zajistí, že produkt IBM MQ převede data během přenosu. V tomto případě musí být uživatelská procedura pro převod dat umístěna v systému, kde je definován kanál odesilatele.

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 `CodedCharSetId` na MQCCSI_Q_MGR a `Encoding` na MQENC_NATIVE. Po dokončení volání MQGET mají tato pole hodnoty odpovídající vráceným datům zprávy. Tyto hodnoty se mohou lišit od požadovaných hodnot, pokud konverze nebyla úspěšná. Aplikace by měla tato pole resetovat na hodnoty požadované před každým voláním MQGET.

Podmínky vyžadované pro volání uživatelské procedury pro převod dat jsou definovány pro volání MQGET v příkazu MQGET.

Popis parametrů předaných uživatelské proceduře pro převod dat a podrobné poznámky o použití viz Převod dat pro volání MQ_DATA_CONV_EXIT a strukturu MQDXP.

Programy, které převádějí data aplikací mezi různými strojovými kódováními a identifikátory CCSID, musí odpovídat rozhraní DCI (IBM MQ data conversion interface).

Pro klienty výběrového vysílání je třeba, aby uživatelské procedury rozhraní API a uživatelské procedury pro převod dat mohly být spuštěny na straně klienta, protože některé zprávy nemusí procházet správcem front. Následující knihovny jsou součástí balíků klienta i balíků serveru:

Tabulka 143. Knihovny, které jsou v balících klienta a serveru

Operační systém	**Knihovny**
AIX	32bitový & 64bitový: libmqm.a & libmqm_r.a
IBM i	LIBMQM & LIBMQM_R
Linux	32bitový & 64bitový: libmqm.so & libmqm_r.so
Windows	32bitový & 64bitový: mqm.dll & mqm.pdb

Vyvolání uživatelské procedury pro převod dat

Uživatelská procedura pro převod dat je uživatelská procedura, která přijímá řízení během zpracování volání MQGET.

Uživatelská procedura je vyvolána v případě, že jsou splněny následující podmínky:

- Volba MQGMO_CONVERT je určena ve volání MQGET.
- Některá nebo všechna data zprávy nejsou v požadované znakové sadě nebo kódování.
- Pole *`Format`* ve struktuře MQMD přidružené ke zprávě není MQFMT_NONE.

- Hodnota *BufferLength* určená ve volání MQGET není nula.
- Délka dat zprávy není nula.
- Zpráva obsahuje data, která mají uživatelsky definovaný formát. Uživatelem definovaný formát může obsadit celou zprávu nebo mu může předcházet jeden nebo více vestavěných formátů. Před formátem definovaným uživatelem může například předcházet formát MQFMT_DEAD_LETTER_HEADER. Uživatelská procedura je vyvolána za účelem převodu pouze formátu definovaného uživatelem; správce front převádí všechny vestavěné formáty, které předcházejí formátu definovanému uživatelem.

  Uživatelskou proceduru zapsanou uživatelem lze také vyvolat pro převod vestavěného formátu, ale k tomu dochází pouze v případě, že vestavěné převodní rutiny nemohou úspěšně převést vestavěný formát.

Existují některé další podmínky, které jsou plně popsány v poznámkách k použití volání MQ_DATA_CONV_EXIT v MQ_DATA_CONV_EXIT.

Podrobnosti o volání MQGET viz MQGET . Uživatelské procedury pro převod dat nemohou používat jiná volání MQI než MQXCNVC.

Nová kopie uživatelské procedury se načte, když se aplikace pokusí načíst první zprávu, která používá tento soubor *Format* od připojení aplikace ke správci front. Nová kopie může být také načtena jindy, pokud správce front zrušil dříve načtenou kopii.

Uživatelská procedura převodu dat je spuštěna v prostředí, které se podobá prostředí programu, který zadal volání MQGET. Kromě uživatelských aplikací může program představovat agenta MCA (agent kanálu zpráv) odesílající zprávy do cílového správce front, který nepodporuje převod zpráv. Prostředí zahrnuje adresní prostor a profil uživatele, kde je to možné. Uživatelská procedura nemůže ohrozit integritu správce front, protože není spuštěna v prostředí správce front.

Převod dat na z/OS

z/OS

V systému z/OSmějte na paměti následující skutečnosti:

- Uživatelské programy mohou být napsány pouze v jazyce sestavení.
- Uživatelské programy musí být reentrant a musí být schopné pracovat kdekoli v úložišti.
- Uživatelské programy musí obnovit prostředí při ukončení na to při vstupu a musí uvolnit jakoukoli získanou paměť.
- Uživatelské programy nesmí WAIT ani vydávat ESTAEs nebo SPIEs.
- Uživatelské programy jsou obvykle vyvolávány pomocí z/OS LINK v:
  - Neautorizovaný problémový stav programu
  - Režim řízení primárního adresního prostoru
  - Režim bez křížové paměti
  - Režim bez přístupu-registrace
  - 31bitový režim adresování
  - Režim TCB-PRB
- Při použití aplikací CICS je uživatelská procedura vyvolána pomocí EXEC CICS LINK a musí být v souladu s konvencemi programování CICS . Parametry jsou předávány ukazateli (adresami) v komunikační oblasti CICS (COMMAREA).

  Ačkoli to není doporučeno, uživatelské programy mohou také používat volání rozhraní API CICS s následujícím upozorněním:

  - Nevydejte synchronizační body, protože výsledky by mohly ovlivnit jednotky práce deklarované agentem MCA.
  - Neaktualizujte žádné prostředky řízené jiným správcem prostředků než IBM MQ for z/OS, včetně prostředků řízených serverem CICS Transaction Server.

Pro kanály s hodnotou CONVERT = YES je uživatelská procedura načtena z datové sady, na kterou odkazuje příkaz CSQXLIB DD. MQ-dodané uživatelské procedury CSQCBDCI a CSQCBDCO pro most IBM MQ CICS jsou v SCSQAUTH.

Zápis uživatelského programu pro převod dat pro IBM i

Informace o krocích, které je třeba zvážit při zápisu programů uživatelské procedury pro převod dat produktu MQ pro produkt IBM i.

Postupujte takto:

1. Pojmenujte svůj formát zprávy. Název se musí vejít do pole *Format* deskriptoru MQMD. Název *Format* nesmí mít úvodní vložené mezery a koncové mezery jsou ignorovány. Název objektu nesmí mít více než osm neprázdných znaků, protože *Format* má pouze osm znaků. Nezapomeňte použít toto jméno pokaždé, když odešlete zprávu (náš příklad používá název Format).
2. Vytvořte strukturu, která bude představovat vaši zprávu. Příklad viz Platná syntaxe .
3. Spuštěním této struktury prostřednictvím příkazu CVTMQMDTA vytvořte fragment kódu pro uživatelskou proceduru převodu dat.

   Funkce generované příkazem CVTMQMDTA používají makra dodávaná v souboru QMQM/H (AMQSVMHA). Tato makra jsou napsána za předpokladu, že všechny struktury jsou zabaleny; pokud tomu tak není, byla pozměněna.
4. Vytvořte kopii dodaného zdrojového souboru kostry QMQMSAMP/QCSRC (AMQSVFC4) a přejmenujte jej. (Náš příklad používá název EXIT_MOD.)
5. Ve zdrojovém souboru vyhledejte následující pole s komentáři a vložte kód podle popisu:

   a. Na konci zdrojového souboru začíná rámeček s komentářem:

   ```
   /* 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 tomto volání následuje volání funkce ConverttagSTRUCTs komentářem.

   Změňte název funkce na název funkce, kterou jste přidali v kroku “5.a” na stránce 953. Chcete-li funkci aktivovat, 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 953.

   Pokud zpráva obsahuje znaková data, generovaný kód volá MQXCNVC; to lze vyřešit vazbou servisního programu QMQM/LIBMQM.
6. Kompilujte zdrojový modul EXIT_MOD následujícím způsobem:

   ```
   CRTCMOD MODULE(library/EXIT_MOD) +
   SRCFILE(QCSRC) +
   TERASPACE(*YES *TSIFC)
   ```
7. Vytvořte/propojte program.

   V případě aplikací bez podprocesů 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 se v prostředí s podporou podprocesů požaduje další. Za tímto načteným objektem musí následovat _R. Pomocí knihovny LIBMQM_R vyřešte volání modulu MQXCNVC. Oba objekty s možností načtení jsou vyžadovány pro prostředí s podporou podprocesů.

```
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 uživatelské programy pro konverzi dat uloženy v knihovně QSYS.

Poznámka:

1. Pokud CVTMQMDTA používá sbalené struktury, všechny aplikace IBM MQ musí používat kvalifikátor _Packed.
2. Uživatelské programy pro převod dat musí být reentrant.
3. MQXCNVC je jediné volání MQI, které lze zadat z uživatelské procedury pro převod dat.
4. Kompilujte uživatelský program s volbou kompilátoru uživatelského profilu nastavenou na *USER, aby se uživatelská procedura spustila s oprávněním uživatele.
5. Povolení paměti Teraspace je vyžadováno pro všechny uživatelské procedury s produktem IBM MQ for IBM i ; Zadejte TERASPACE (*YES *TSIFC) v příkazech CRTCMOD a CRTBNDC.

Zápis uživatelského programu pro převod dat pro IBM MQ for z/OS

Informace o krocích, které je třeba zvážit při zápisu programů uživatelské procedury pro převod dat pro IBM MQ for z/OS.

Postupujte takto:

1. Použijte dodanou zdrojovou kostru CSQ4BAX9 (pro jiná prostředí nežCICS) nebo CSQ4CAX9 (pro CICS) jako váš výchozí bod.
2. Spusťte obslužný program CSQUCVX.
3. Podle pokynů v prologu CSQ4BAX9 nebo CSQ4CAX9 začleňte rutiny generované obslužným programem CSQUCVX v pořadí, v jakém se struktury vyskytují ve zprávě, kterou chcete převést.
4. Obslužný program předpokládá, že datové struktury nejsou komprimovány, že je respektováno implicitní zarovnání dat a že struktury začínají na hranici celého slova, přičemž bajty jsou podle potřeby vynechány (jako mezi ID a VERZE v příkladu v platné syntaxi). Jsou-li struktury komprimovány, vynechte generovaná makra CMQXCALA. Proto zvažte deklarování svých struktur takovým způsobem, aby byla všechna pole pojmenována a nebyly vynechány žádné bajty; v příkladu v Platná syntaxepřidejte pole "MQBYTE DUMMY;" mezi ID a VERSION.
5. Zadaná uživatelská procedura vrátí chybu, pokud je vstupní vyrovnávací paměť kratší než formát zprávy, která má být převedena. Přestože uživatelská procedura převede co nejvíce úplných polí, chyba způsobí vrácení nepřevedené zprávy do aplikace. Chcete-li povolit co největší převod krátkých vstupních vyrovnávacích pamětí, včetně dílčích polí, změňte hodnotu TRUNC= v makru CSQXCDFA na YES: nevrátí se žádná chyba, takže aplikace obdrží převedenou zprávu. Aplikace musí oříznutí zpracovat.
6. Přidejte další speciální kód zpracování, který potřebujete.
7. Přejmenujte program na název datového formátu.
8. Zkompilujte a propojte-upravte svůj program jako dávkový aplikační program (pokud není určen pro použití s aplikacemi CICS). Makra v kódu generovaném obslužným programem jsou v knihovně **thlqual**.SCSQMACS.

Pokud zpráva obsahuje znaková data, generovaný kód volá MQXCNVC. Pokud vaše uživatelská procedura používá toto volání, upravte jej pomocí programu výstupního stubu CSQASTUB. Stub je nezávislý na jazyku a nezávislý na prostředí. Alternativně můžete stub načíst dynamicky pomocí dynamického názvu volání CSQXCNVC. Další informace viz “Dynamické volání stubu IBM MQ” na stránce 995.

Umístěte modul upravený odkazem do knihovny načtení aplikace a do datové sady, na kterou odkazuje příkaz CSQXLIB DD vaší procedury úlohy spuštěné vaším inicializátorem kanálu.

9. Pokud je uživatelská procedura určena pro použití aplikacemi CICS , kompilujte ji a upravte ji pomocí odkazů jako aplikační program CICS , v případě potřeby včetně CSQASTUB. Umístěte jej do knihovny aplikačního programu CICS . Typickým způsobem definujte program pro CICS a zadejte EXECKEY (CICS) v definici.

Poznámka: Ačkoli jsou běhové knihovny LE/370 potřebné pro spuštění obslužného programu CSQUCVX (viz krok “2” na stránce 954), nejsou potřebné pro linkování nebo spuštění uživatelské procedury pro převod dat (viz kroky “8” na stránce 954 a “9” na stránce 955).

Informace o převodu dat v rámci mostu IBM MQ - IMS naleznete v části “Zápis aplikací mostu IMS” na stránce 70 .

Linux AIX *Zápis uživatelské procedury pro převod dat pro systémy IBM MQ for AIX or Linux*

Informace o krocích, které je třeba zvážit při zápisu programů uživatelské procedury pro převod dat pro systémy IBM MQ for AIX or Linux .

Postupujte takto:

1. Pojmenujte svůj formát zprávy. Název se musí vejít do pole *Format* deskriptoru MQMD a musí být uveden velkými písmeny, například MYFORMAT. Název *Format* nesmí mít úvodní mezery. Koncové mezery jsou ignorovány. Název objektu nesmí mít více než osm neprázdných znaků, protože *Format* má pouze osm znaků. Nezapomeňte použít toto jméno pokaždé, když odešlete zprávu.

   Pokud se uživatelská procedura převodu dat používá v prostředí s podporou podprocesů, musí být zaveditelný objekt následován _r, aby označil, že se jedná o verzi s podporou podprocesů.

2. Vytvořte strukturu, která bude představovat vaši zprávu. Příklad viz Platná syntaxe .
3. Spuštěním této struktury prostřednictvím příkazu `crtmqcvx` vytvořte fragment kódu pro uživatelskou proceduru převodu dat.

   Funkce generované příkazem `crtmqcvx` používají makra, která předpokládají, že jsou sbaleny všechny struktury; pokud tomu tak není, změňte je.

4. Zkopírujte dodaný zdrojový soubor kostry a přejmenujte jej na název formátu zprávy, který jste nastavili v kroku “1” na stránce 955. Zdrojový soubor kostry a kopie jsou jen pro čtení.

   Zdrojový soubor kostry se nazývá amqsvfc0.c.

5. V systému IBM MQ for AIXje také dodán soubor exportu kostry s názvem amqsvfc.exp . Zkopírujte tento soubor a přejmenovte jej na MYFORMAT.EXP.
6. Kostra zahrnuje 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, abyste tento soubor vyzvedli.

   Soubor amqsvmha.h obsahuje makra, která jsou použita kódem generovaným příkazem `crtmqcvx` . Pokud struktura, která má být převedena, obsahuje znaková data, tato makra volají MQXCNVC.

7. Ve zdrojovém souboru vyhledejte následující pole s komentáři a vložte kód podle popisu:

   a. Na konci zdrojového souboru začíná rámeček s komentářem:

   ```
   /* Insert the functions produced by the data-conversion exit */
   ```

   Zde vložte fragment kódu vygenerovaný v kroku “3” na stránce 955.

b. V blízkosti středu zdrojového souboru začíná rámeček s komentářem:

```
/* Insert calls to the code fragments to convert the format's */
```

Po tomto volání následuje volání funkce `ConverttagSTRUCTs` komentářem.

Změňte název funkce na název funkce, kterou jste přidali v kroku “7.a” na stránce 955. Chcete-li funkci aktivovat, 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 955.

8. Zkompilujte uživatelskou proceduru jako sdílenou knihovnu s použitím MQStart jako vstupního bodu. Chcete-li to provést, prohlédněte si téma “Kompilace uživatelských procedur pro převod dat v systémech AIX and Linux” na stránce 956.
9. Umístěte výstup do adresáře ukončení. Výchozí adresář uživatelské procedury je `/var/mqm/exits` pro 32bitové systémy a `/var/mqm/exits64`pro 64bitové systémy. Tyto adresáře můžete změnit v souboru qm.ini nebo mqclient.ini . Tuto cestu lze nastavit pro každého správce front a uživatelská procedura je hledána pouze v této cestě nebo cestách.

Poznámka:

1. Pokud produkt `crtmqcvx` používá sbalené struktury, musí být všechny aplikace IBM MQ kompilovány tímto způsobem.
2. Uživatelské programy pro převod dat musí být reentrant.
3. MQXCNVC je jediné volání MQI, které lze zadat z uživatelské procedury pro převod dat.

Linux AIX *Kompilace uživatelských procedur pro převod dat v systémech AIX and Linux*

Příklady, jak kompilovat uživatelskou proceduru pro převod dat na systémech AIX and Linux .

Na všech platformách je vstupní bod modulu MQStart.

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

AIX

AIX

Kompilujte zdrojový kód ukončení zadáním jednoho z následujících příkazů:

32bitové aplikace

Bez podprocesů

```
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

Bez podprocesů

```
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
```

Linux

Linux

Kompilujte zdrojový kód ukončení zadáním jednoho z následujících příkazů:

31 bitové aplikace

Bez podprocesů

```
 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

Bez podprocesů

```
 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

Bez podprocesů

```
 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
```

Windows Zápis uživatelské procedury pro převod dat pro IBM MQ for Windows

Informace o krocích, které je třeba zvážit při zápisu programů uživatelské procedury pro převod dat pro IBM MQ for Windows.

Postupujte takto:

1. Pojmenujte svůj formát zprávy. Název se musí vejít do pole *Format* deskriptoru MQMD. Název *Format* nesmí mít úvodní mezery. Koncové mezery jsou ignorovány. Název objektu nesmí mít více než osm neprázdných znaků, protože *Format* má pouze osm znaků.

   Soubor .DEF s názvem amqsvfcn.def je také dodáván 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 vytvářené knihovny DLL a název uvedený v MYFORMAT.DEF jsou stejné. Přepište název FORMAT1 v MYFORMAT.DEF s novým názvem formátu.

   Nezapomeňte použít toto jméno pokaždé, když odešlete zprávu.
2. Vytvořte strukturu, která bude představovat vaši zprávu. Příklad viz Platná syntaxe .
3. Spuštěním této struktury prostřednictvím příkazu `crtmqcvx` vytvořte fragment kódu pro uživatelskou proceduru převodu dat.

   Funkce generované příkazem CRTMQCVX používají makra, která jsou napsána za předpokladu, že jsou zabaleny všechny struktury; pokud tomu tak není, změňte je.
4. Zkopírujte dodaný zdrojový soubor kostry amqsvfc0.ca přejmenujte jej na název vašeho formátu zprávy, který jste nastavili v kroku “1” na stránce 958.

   amqsvfc0.c se nachází v adresáři *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, abyste tento soubor vyzvedli.

   Soubor amqsvmha.h obsahuje makra, která jsou použita kódem generovaným příkazem CRTMQCVX. Pokud struktura, která má být převedena, obsahuje znaková data, tato makra volají MQXCNVC.
5. Ve zdrojovém souboru vyhledejte následující pole s komentáři a vložte kód podle popisu:

   a. Na konci zdrojového souboru začíná rámeček s komentářem:

   ```
   /* Insert the functions produced by the data-conversion exit */
   ```

   Zde vložte fragment kódu vygenerovaný v kroku “3” na stránce 958.

   b. V blízkosti středu zdrojového souboru začíná rámeček s komentářem:

   ```
   /* Insert calls to the code fragments to convert the format's */
   ```

   Po tomto volání následuje volání funkce `Converttag`STRUCTs komentářem.

   Změňte název funkce na název funkce, kterou jste přidali v kroku “5.a” na stránce 958. Chcete-li funkci aktivovat, 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 958.

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. Zadáním příkazového souboru zkompilujte uživatelskou proceduru jako soubor DLL.
8. Umístěte výstup do podadresáře exit pod datový adresář IBM MQ . Výchozí adresář pro instalaci 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. Tuto cestu lze nastavit pro každého správce front a uživatelská procedura je hledána pouze v této cestě nebo cestách.

Poznámka:

1. Pokud CRTMQCVX používá sbalené struktury, všechny aplikace IBM MQ musí být kompilovány tímto způsobem.
2. Uživatelské programy pro převod dat musí být reentrant.
3. MQXCNVC je jediné volání MQI, které lze zadat z uživatelské procedury pro převod dat.

Windows *Ukončete a přepněte soubory načtení na operačních systémech Windows .*

Procesy správce front IBM WebSphere MQ for Windows 7.5 jsou 32bitové. V důsledku toho při použití 64bitových aplikací musí mít některé typy uživatelských procedur a souborů načtení přepínače XA také k dispozici 32bitovou verzi pro použití správcem front. Pokud je vyžadována 32bitová verze souboru ukončení nebo načtení přepínače XA a není k dispozici, dojde k selhání příslušného volání rozhraní API nebo příkazu.

V adresáři qm.ini file pro *ExitPath*jsou podporovány dva atributy. 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 . Použitím těchto prostředků zajistíte, že bude nalezena příslušná knihovna. Pokud se uživatelská procedura používá v klastru IBM MQ , zajistí se také, že bude na vzdáleném systému nalezena příslušná knihovna.

V následující tabulce jsou uvedeny různé typy souborů načtení Exit and Switch a poznámky, zda jsou vyžadovány 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	32bitové a 64bitové	64bitová
Uživatelská procedura převodu dat	32bitové	64bitová
Ukončení kanálu serveru (všechny typy)	64bitová	64bitová

typy souboru	32bitové aplikace	64bitové aplikace
Uživatelské procedury kanálu klienta (všechny typy)	32bitové	64bitová
Uživatelská procedura instalovatelné služby	64bitová	64bitová
Ukončení správy WLM klastru	64bitová	64bitová
Uživatelská procedura směrování publikování/odběru	64bitová	64bitová
Soubory načtení přepínače databáze	32bitové a 64bitové	64bitová
Externí knihovny AX správce transakcí	32bitové	64bitová
Ukončení před připojením	32bitové	64bitová

Odkazování na definice připojení pomocí uživatelské procedury před připojením z úložiště

Produkt IBM MQ MQI clients lze nakonfigurovat tak, aby vyhledal úložiště a získal definice připojení pomocí knihovny uživatelské procedury před připojením.

Úvod

Aplikace klienta se může připojit ke správci front pomocí tabulek CCDT (Client Channel Definition Table). Obecně je soubor CCDT umístěn na centrálním síťovém souborovém serveru a klienti na něj 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, registr a úložiště WebSphere nebo jiné úložiště. Uložení definic připojení klienta do úložiště 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 knihovnu uživatelské procedury před připojením určenou aplikací a vyvolá funkci uživatelské procedury pro načtení definic připojení. Načtené definice připojení se poté použijí k vytvoření připojení ke správci front. Podrobnosti o knihovně uživatelské procedury a funkci, která má být vyvolána, jsou určeny v konfiguračním souboru mqclient.ini .

Syntaxe

`void MQ_PRECONNECT_EXIT` (*pExitParms*, *pQMgrNázev*, *ppConnectOpts*, *pCompKód*, *pReason*);

Parametry

pExitParms (parametry)

Typ: PMQNXP vstup/výstup

Struktura parametru ukončení **`PreConnection`** .

Struktura je přidělena a udržována volajícím na výstupu.

Název pQMgr

Typ: Vstupní/výstup typu PMQCHAR

Název správce front.

Na vstupu je tento parametr řetězec filtru dodaný do volání rozhraní API MQCONN prostřednictvím parametru **`QMgrName`** . Toto pole může být prázdné, explicitní nebo může obsahovat určité zástupné znaky. Pole se změní uživatelskou procedurou. Při volání uživatelské procedury s MQXR_TERM má parametr hodnotu NULL.

ppConnectvolby
Typ: ppConnectVstupní/výstupní volby

Volby, které řídí akci MQCONNX.

Jedná se o ukazatel na strukturu voleb připojení MQCNO, která řídí akci volání rozhraní API MQCONN. Při volání uživatelské procedury s MQXR_TERM má parametr hodnotu NULL. Klient MQI vždy poskytuje uživatelskou proceduru strukturu MQCNO, a to i v případě, že nebyla původně poskytnuta aplikací. Pokud aplikace poskytuje strukturu MQCNO, klient vytvoří duplikát, aby ji předal uživatelské proceduře, kde je upravena. Klient si ponechává vlastnictví MQCNO.

MQCD odkazované prostřednictvím MQCNO má přednost před jakoukoli definicí připojení poskytnutou prostřednictvím pole. Klient používá strukturu MQCNO pro připojení ke správci front a ostatní jsou ignorovány.

Kód pComp
Typ: Vstupní/výstupní PMQLONG

Kód dokončení.

Ukazatel na objekt MQLONG, který přijímá kód dokončení ukončení. Musí to být jedna z následujících hodnot:

- MQCC_OK -Úspěšné dokončení
- MQCC_WARNING -Varování (částečné dokončení)
- MQCC_FAILED -Volání se nezdařilo.

pReason
Typ: Vstupní/výstupní PMQLONG

Kód příčiny pComp.

Ukazatel na MQLONG, který obdrží kód příčiny ukončení. Pokud je kód dokončení MQCC_OK, jediná platná hodnota je:

- MQRC_NONE-(0, x '000') Není důvod k hlášení.

Pokud je kód dokončení MQCC_FAILED nebo MQCC_WARNING, funkce ukončení může nastavit pole kódu příčiny na libovolnou platnou hodnotu MQRC_ *.

Vyvolání 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 publikování

Uživatelskou proceduru publikování ve správci front můžete nakonfigurovat tak, aby před přijetím publikované zprávy odběrateli změnila obsah publikované zprávy. Můžete také změnit záhlaví zprávy nebo nedoručit zprávu do odběru.

Poznámka: Uživatelské procedury publikování nejsou v systému z/OSpodporovány.

Uživatelskou proceduru publikování můžete použít ke kontrole a změně zpráv doručených odběratelům:

- Zkontrolovat obsah zprávy publikované pro každého odběratele
- Upravit obsah zprávy publikované každému odběrateli

- Změnit frontu, do které je zpráva vložena
- Zastavit doručování zprávy odběrateli

Zápis uživatelské procedury publikování

Použijte kroky v části “Zápis uživatelských procedur a instalovatelných služeb na webu AIX, Linux, and Windows” na stránce 904, které vám pomohou napsat a zkompilovat uživatelskou proceduru.

Poskytovatel uživatelské procedury publikování definuje, co uživatelská procedura provede. Uživatelská procedura však musí být v souladu s pravidly definovanými v MQPSXP.

IBM MQ neposkytuje implementaci vstupního bodu MQ_PUBLISH_EXIT. Poskytuje deklaraci typedef jazyka C. Pomocí funkce typedef deklarujte parametry správně uživatelské proceduře zapsané uživatelem. 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 spouští v rámci procesu správce front v důsledku následujících operací:

- Operace publikování, při které je zpráva doručena jednomu nebo více odběratelům.
- Operace odběru, při které je doručena jedna nebo více uchovaných zpráv.
- Operace požadavku na odběr, při které je doručena jedna nebo více uchovaných zpráv

Je-li pro připojení volána uživatelská procedura publikování, je při prvním volání nastavena hodnota *ExitReason* kód MQXR_INIT . Před odpojením připojení 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, budou pro správce front blokovány operace publikování/odběru zpráv. Před opětovným povolením systému zpráv publikování/odběru je třeba opravit problém nebo restartovat správce front.

Každé připojení IBM MQ , které vyžaduje uživatelskou proceduru publikování, se nemusí podařit načíst nebo inicializovat uživatelskou proceduru. Pokud se uživatelské proceduře nepodaří načíst nebo inicializovat, operace publikování/odběru, které vyžadují uživatelskou proceduru publikování, jsou pro toto připojení zakázány. Operace se nezdaří s IBM MQ kódem příčiny 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 uchovat 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. Jedná se o tyto hovory:

- MQBUFMH
- MQCRTMH
- MQDLTMH
- MQDLTMP
- MQMHBUF
- MQINQMP
- MQSETMP

Pokud uživatelská procedura publikování změní název cílového správce front nebo název fronty, nebude provedena žádná nová kontrola oprávnění.

Kompilace uživatelské procedury publikování

Uživatelská procedura publikování je dynamicky načtená knihovna; lze ji považovat za uživatelskou proceduru kanálu. Informace o kompilaci uživatelských procedur viz "Zápis uživatelských procedur a instalovatelných služeb na webu AIX, Linux, and Windows" na stránce 904.

Ukázková uživatelská procedura publikování

Ukázkový uživatelský program se nazývá `amqspse0.c`. Zapíše jinou zprávu do souboru protokolu v závislosti na tom, zda byla uživatelská procedura volána pro operace inicializace, publikování nebo ukončení. Také demonstruje použití pole uživatelské oblasti ukončení pro odpovídající přidělení a uvolnění úložiště.

Konfigurace uživatelských procedur publikování

Chcete-li konfigurovat uživatelskou proceduru publikování, musíte definovat určité atributy.

V systémech Windows a Linux můžete k definování atributů použít průzkumník IBM MQ . Atributy jsou definovány na stránce vlastností správce front v části Publikovat/Odebírat.

Chcete-li nakonfigurovat uživatelskou proceduru publikování v souboru `qm.ini` na systémech AIX and Linux , vytvořte sekci s názvem `PublishSubscribe`. Sekce `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 kód uživatelské procedury publikování. Maximální délka tohoto pole je `MQ_EXIT_NAME_LENGTH`.

IBM i V systému IBM i, pokud se používá program, vynechte `PublishExitFunction`.

PublishExitData = *řetězec*
Pokud správce front volá uživatelskou proceduru publikování, předá strukturu `MQPSXP` jako vstup. Data uvedená pomocí atributu **PublishExitData** jsou poskytnuta 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 upravit správu pracovní zátěže klastrů, napište uživatelský program pracovní zátěže klastru. Při směrování zpráv můžete vzít v úvahu náklady na používání kanálu v různých denních časech nebo obsah zpráv. Jedná se o faktory, které nejsou brány v úvahu pomocí standardního algoritmu správy pracovní zátěže.

Ve většině případů je algoritmus správy pracovní zátěže dostačující pro vaše potřeby. Avšak, abyste mohli poskytnout svůj vlastní uživatelský program pro přizpůsobení správy pracovní zátěže, produkt IBM MQ zahrnuje uživatelskou proceduru, uživatelskou proceduru pracovní zátěže klastru.

Můžete mít k dispozici určité konkrétní informace o síti nebo zprávy, které můžete použít k ovlivnění vyrovnávání pracovní zátěže. Můžete vědět, které jsou vysokokapacitní kanály nebo levné síťové trasy, nebo můžete chtít směrovat zprávy v závislosti na jejich obsahu. Můžete se rozhodnout napsat uživatelský program pracovní zátěže klastru nebo použít program dodaný třetí stranou.

Uživatelská procedura pracovní zátěže klastru je volána při přístupu do fronty klastru. Volá jej `MQOPEN`, `MQPUT1` a `MQPUT`.

Je-li zadán parametr `MQOO_BIND_ON_OPEN` , bude opraven cílový správce front vybraný v `MQOPEN` čase. V tomto případě se uživatelská procedura spustí pouze jednou.

Není-li cílový správce front opraven v čase `MQOPEN` , je cílový správce front vybrán v době volání funkce `MQPUT` . Není-li cílový správce front k dispozici nebo dojde-li k selhání v době, kdy je zpráva stále

v přenosové frontě, bude uživatelská procedura znovu volána. Je vybrán nový cílový správce front. Pokud během přenosu zprávy dojde k selhání kanálu zpráv a zpráva je vrácena zpět, je vybrán nový cílový správce front.

Multi V systému Multiplatformsnačte správce front novou uživatelskou proceduru pracovní zátěže klastru při příštím spuštění správce front.

Pokud definice správce front neobsahuje název uživatelského programu pracovní zátěže klastru, nebude uživatelská procedura pracovní zátěže klastru volána.

Do uživatelské procedury pracovní zátěže klastru se ve struktuře parametrů uživatelské procedury MQWXPpředávají různá data:

- Struktura definice zprávy MQMD.
- Parametr délky zprávy.
- Kopie zprávy nebo její části.

Pokud na jiných platformách nežz/OS používáte produkt CLWLMode=FAST, každý proces operačního systému načte svou vlastní kopii uživatelské procedury. Různá připojení ke správci front mohou způsobit vyvolání různých kopií uživatelské procedury. Pokud je uživatelská procedura spuštěna ve výchozím bezpečném režimu CLWLMode=SAFE, spustí se jedna kopie uživatelské procedury ve vlastním odděleném procesu.

Zápis uživatelských procedur pracovní zátěže klastru

z/OS Informace o zápisu uživatelských procedur pracovní zátěže klastru pro z/OSviz “Programování uživatelské procedury pracovní zátěže klastru pro IBM MQ for z/OS” na stránce 966.

V produktu IBM MQ 9.1.0je pracovní zátěž klastru spuštěna v adresním prostoru inicializátoru kanálu namísto adresního prostoru správce front. Máte-li uživatelskou proceduru pracovní zátěže klastru, měli byste odebrat příkaz CSQXLIB DD z procedury spuštěné úlohy správce front a přidat datovou sadu obsahující uživatelskou proceduru pracovní zátěže klastru do zřetězení CSQXLIB v proceduře spuštěné úlohy inicializátoru kanálu.

Multi V případě platformy Multiplatforms nesmí uživatelské procedury pracovní zátěže klastru používat volání MQI. V jiných ohledech jsou pravidla pro zápis a kompilaci programů uživatelských procedur pracovní zátěže klastru podobná pravidlům, která platí pro programy uživatelských procedur kanálu. Postupujte podle pokynů v části “Zápis uživatelských procedur a instalovatelných služeb na webu AIX, Linux, and Windows” na stránce 904a použijte ukázkový program “Uživatelská procedura pracovní zátěže ukázkového klastru” na stránce 964 , který vám pomůže napsat a zkompilovat vaši uživatelskou proceduru.

Další informace o uživatelských procedur kanálu viz “Psaní programů uživatelské procedury kanálu” na stránce 931.

Konfigurace uživatelských procedur pracovní zátěže klastru

Uživatelské procedury pracovní zátěže klastru v definici správce front pojmenujete zadáním atributu uživatelské procedury pracovní zátěže klastru v příkazu ALTER QMGR . Příklad:

```
ALTER QMGR CLWLEXIT(myexit)
```

Související odkazy

Volání ukončení pracovní zátěže klastru a datové struktury

Uživatelská procedura pracovní zátěže ukázkového klastru

Produkt IBM MQ zahrnuje ukázkový uživatelský program pracovní zátěže klastru. Ukázku můžete zkopírovat a použít ji jako základ pro své vlastní programy.

IBM MQ for z/OS

Ukázkový uživatelský program pracovní zátěže klastru je dodáván v modulu Assembler a v adresáři C. Verze asembleru 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 kvalifikátor vyšší úrovně cílové knihovny pro datové sady IBM MQ ve vaší instalaci.

IBM MQ for Multiplatforms

Ukázkový uživatelský program pracovní zátěže klastru je dodáván v jazyce C a nazývá se `amqswlm0.c`. To lze nalézt v:

Tabulka 144. Umístění vzorového uživatelského programu pracovní zátěže klastru pro produkt Multiplatforms

Platforma	**Cesta k souboru**
AIX	*MQ_INSTALLATION_PATH*`/samp`
Windows	*MQ_INSTALLATION_PATH* `\Tools\c\Samples`
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 na konkrétního správce front, pokud se tento správce front nestane nedostupným. Reaguje na selhání správce front směrováním zpráv na jiného správce front.

Určete, kterému správci front mají být odesílány zprávy. Do atributu `CLWLDATA` v definici správce front zadejte název přijímacího kanálu klastru. Příklad:

```
ALTER QMGR CLWLDATA(' my-cluster-name. my-queue-manager ')
```

Chcete-li povolit uživatelskou proceduru, zadejte její úplnou cestu a název do atributu `CLWLEXIT` :

V systému AIX and Linux:

```
ALTER QMGR CLWLEXIT(' path /amqswlm(clwlFunction)')
```

V systému Windows:

```
ALTER QMGR CLWLEXIT(' path \amqswlm(clwlFunction)')
```

V systému z/OS:

```
ALTER QMGR CLWLEXIT(CSQ4BxF1)
```

kde x je buď `'A'` , nebo `'C'`, v závislosti na programovacím jazyku verze, kterou používáte.

V systému IBM ipoužijte jeden z následujících příkazů:

- Použijte příkaz MQSC:

```
ALTER QMGR CLWLEXIT('AMQSWLM       library       ')
```

Název programu i název knihovny zabírají 10 znaků a jsou v případě potřeby vyplněny mezerou vpravo.

- Použijte příkaz CL:

```
CHGMQM MQMNAME( qmgrname ) CLWLEXIT(' library /AMQSWLM')
```

Nyní produkt IBM MQ namísto použití dodaného algoritmu správy pracovní zátěže volá tuto uživatelskou proceduru, aby směrovala všechny zprávy do vybraného správce front.

z/OS *Programování uživatelské procedury pracovní zátěže klastru pro IBM MQ for z/OS*

Uživatelské procedury pracovní zátěže klastru jsou vyvolávány jako příkaz z/OS **LINK** . Východy podléhají řadě přísných programových pravidel. Vyvarujte 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 je vyvolal z/OS **LINK** v:

- Neautorizovaný problémový stav programu
- Režim řízení primárního adresního prostoru
- Nekřížový režim paměti
- Režim registru bez přístupu
- 31bitový režim adresování
- Klíč úložiště 8
- Maska klíče programu 8
- Klíč TCB 8

Vložte moduly upravené odkazem do datové sady určené příkazem CSQXLIB DD procedury spuštěné úlohy inicializátoru kanálu. 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:

- Uživatelské procedury musíte zapsat v assembleru nebo C. Používáte-li jazyk C, musí být v souladu s programovacím prostředím systémů C pro systémové uživatelské procedury, jak je popsáno v příručce *z/OS C/C++ Programming Guide*, SC09-4765.
- Používáte-li volání MQXCLWLN , propojte úpravu s CSQMFCLW, dodanou v *thlqual*.SCSQLOAD.
- Uživatelské procedury jsou načteny z neautorizovaných knihoven definovaných příkazem CSQXLIB DD . Pokud má parametr CSQXLIB k dispozici DISP=SHR, lze uživatelské procedury aktualizovat za běhu správce front s použitím nové verze použité v dalším podprocesu produktu MQCONN , který spustí správce front.
- Uživatelské procedury musí být reentrant a musí být schopné pracovat kdekoli ve virtuálním úložišti.
- Uživatelské procedury musí resetovat prostředí při návratu k prostředí při vstupu.
- Uživatelské procedury musí uvolnit jakoukoli získanou paměť, nebo se musí ujistit, že je paměť uvolněna následným vyvoláním ukončení.
- Nejsou povolena žádná volání MQI.
- Uživatelské procedury nesmí používat žádné systémové služby, které by mohly způsobit čekání, protože čekání výrazně snižuje výkon správce front. Obecně tedy vyvarujte se SVC, PC nebo I/O.
- Uživatelské procedury nesmí vydávat ESTAE nebo SPIE, kromě dílčích úloh, které připojují.

Poznámka: Neexistují žádná absolutní omezení týkající se toho, co můžete dělat ve výjezdu. Většina okruhů SVC však vyžaduje čekání, takže se jim vyhýbejte, s výjimkou následujících příkazů:

- **GETMAIN** / **FREEMAIN**
- **LOAD** / **DELETE**

Nepoužívat ESTAEs a ESPIEs, protože jejich ošetřování chyb může kolidovat s ošetřování chyb prováděným produktem IBM MQ. Produkt IBM MQ nemusí být schopen se zotavit z chyby, nebo váš uživatelský program neobdrží všechny informace o chybě.

Systémový parametr EXITLIM omezuje dobu, po kterou může uživatelská procedura běžet. Výchozí hodnota pro EXITLIM je 30 sekund. Pokud uvidíte návratový kód MQRC_CLUSTER_EXIT_ERROR, 2266 X'8DA' , může být vaše uživatelská procedura ve smyčce. Pokud si myslíte, že uživatelská procedura potřebuje k dokončení více než 30 sekund, zvyšte hodnotu EXITLIM.

Sestavení procedurální aplikace

Aplikaci IBM MQ můžete napsat v jednom z několika procedurálních jazyků a spustit ji na několika různých platformách.

AIX Sestavení vaší procedurální aplikace na AIX

Publikace AIX popisují, jak sestavit spustitelné aplikace z programů, které píšete.

Toto téma popisuje další úlohy a změny standardních úloh, které musíte provést při sestavování aplikací IBM MQ for AIX , které mají být spuštěny v produktu AIX. Jsou podporovány jazyky C, C + + a COBOL. Informace o přípravě programů C++ viz Použití C++.

Úlohy, které musíte provést, chcete-li vytvořit spustitelnou aplikaci pomocí produktu IBM MQ for AIX , se liší podle programovacího jazyka, ve kterém je napsán váš zdrojový kód. Kromě kódování volání MQI ve zdrojovém kódu musíte přidat příslušné jazykové příkazy, aby byly zahrnuty soubory začlenění IBM MQ for AIX pro jazyk, který používáte. Seznamte se s obsahem těchto souborů. Úplný popis naleznete v části “Soubory definice dat IBM MQ” na stránce 696 .

Když spouštíte serverové nebo klientské aplikace s podporou podprocesů, nastavte proměnnou prostředí AIXTHREAD_SCOPE = S.

AIX *Příprava programů v jazyce C v adresáři AIX*

Toto téma obsahuje informace o propojování knihoven nezbytných pro přípravu programů v jazyce C na systému AIX.

Předkompilované programy v jazyce C jsou dodávány v adresáři *MQ_INSTALLATION_PATH*/samp/bin . Použijte kompilátor ANSI a spusťte následující příkazy. Další informace o programování 64bitových aplikací naleznete v tématu Standardy kódování na 64bitových platformách.

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

Pro 32bitové 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.

V 9.3.5 Pro 32bitové aplikace používající kompilátor XLC 17:

```
$ ibm-clang -o amqsput_32 amqsput0.c -I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib
-lmqm
```

kde amqsput0 je ukázkový program.

V 9.3.5 Pro 64bitové aplikace používající kompilátor XLC 17:

```
$ ibm-clang -m64 -o amqsput_64 amqsput0.c -I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/
lib64 -lmqm
```

kde `amqsput0` je ukázkový program.

Používáte-li kompilátor VisualAge C/C++ pro programy C++, musíte zahrnout volbu `-q namemangling=v5` , aby se všechny symboly IBM MQ vyřešily při propojování knihoven.

Chcete-li používat programy v počítači, ve kterém je nainstalován pouze produkt IBM MQ MQI client for AIX , znovu zkompilujte programy, aby je propojily s knihovnou klienta (`-lmqic`).

Propojování knihoven

Potřebujete následující knihovny:

- Propojte své programy s příslušnou knihovnou poskytovanou společností IBM MQ.

  V prostředí bez podprocesů vytvořte odkaz na jednu z následujících knihoven:

Soubor knihovny	Typ programu/ukončení
libmqm.a	Server pro C
libmqic.a & libmqm.a	Klient pro C

  V prostředí s podporou podprocesů vytvořte odkaz na jednu 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 s podporou podprocesů z jedné kompilační jednotky, spusťte následující příkazy.

  Pro 32bitové 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.

  V 9.3.5 Pro 32bitové aplikace používající kompilátor XLC 17:

```
$ ibm-clang_r -o amqsput_32_r amqsput0.c -I MQ_INSTALLATION_PATH/inc -L
MQ_INSTALLATION_PATH/lib -lmqm_r
```

  kde `amqsput0` je ukázkový program.

  V 9.3.5 Pro 64bitové aplikace používající kompilátor XLC 17:

```
$ ibm-clang_r -m64 -o amqsput_64_r amqsput0.c -I MQ_INSTALLATION_PATH/inc -L
MQ_INSTALLATION_PATH/lib64 -lmqm_r
```

  kde `amqsput0` je ukázkový program.

  Chcete-li používat programy v počítači, ve kterém je nainstalován pouze produkt IBM MQ MQI client for AIX , znovu zkompilujte programy, aby je propojily s knihovnou klienta (`-lmqic`).

Poznámka:

1. Nelze vytvořit odkaz na více než jednu knihovnu. To znamená, že nemůžete současně odkazovat na knihovnu s podporou podprocesů i na knihovnu bez podpory podprocesů.
2. Pokud píšete instalovatelnou službu (další informace viz Administrace IBM MQ), musíte se odkázat na knihovnu `libmqmzf.a` v nevláknové aplikaci a na knihovnu `libmqmzf_r.a` v vláknové aplikaci.
3. Pokud produkujete aplikaci pro externí koordinaci pomocí správce transakcí podporujícího standard XA, jako např. IBM TXSeries, Encina nebo BEA Tuxedo, musíte se odkázat na `libmqmxa.a` (nebo `libmqmxa64.a` , pokud váš správce transakcí považuje typ 'long' za 64bitový) a knihovny `libmqz.a` v aplikaci bez podprocesů a na `libmqmxa_r.a` (nebo `libmqmxa64_r.a`). a knihovny `libmqz_r.a` v aplikaci s podporou podprocesů.
4. Musíte propojit důvěryhodné aplikace s knihovnami IBM MQ s podprocesy. V systému IBM MQ for AIX or Linux však lze v daném okamžiku připojit pouze jeden podproces v důvěryhodné aplikaci.
5. Knihovny IBM MQ musíte propojit před ostatními knihovnami produktu.

AIX *Příprava programů v jazyce COBOL v adresáři AIX*

Tyto informace použijte při přípravě programů v jazyce COBOL v produktu AIX pomocí IBM COBOL Set a Micro Focus COBOL.

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

- 32bitové kopírovací knihy COBOL jsou nainstalovány v následujícím adresáři:

  ```
  MQ_INSTALLATION_PATH/inc/cobcpy32
  ```

  a symbolické odkazy jsou vytvořeny v:

  ```
  MQ_INSTALLATION_PATH/inc
  ```

- 64bitové kopírovací knihy COBOL se instalují do následujícího adresáře:

  ```
  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.

Program je třeba propojit s jedním z následujících souborů knihovny:

Soubor knihovny	Typ programu/ukončení
libmqmcb.a	Server pro COBOL (nepodprocesová aplikace)
libmqmcb_r.a	Server pro COBOL (aplikace s podporou podprocesů)
libmqicb.a	Klient pro COBOL (nepodprocesová aplikace)
libmqicb_r.a	Klient pro COBOL (aplikace s podporou podprocesů)

V závislosti na programu můžete použít kompilátor IBM COBOL Set nebo kompilátor Micro Focus COBOL:

- Programy začínající `amqm` jsou vhodné pro kompilátor Micro Focus COBOL a
- Programy začínající `amq0` jsou vhodné pro oba kompilátory.

Příprava programů v jazyce COBOL pomocí IBM sady COBOL pro AIX

Ukázkové programy v jazyce COBOL jsou dodávány s produktem IBM MQ. Chcete-li takový program zkompilovat, 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 aplikace s podporou podprocesů

```
$ cob2_r -o amq0put0 amq0put0.cbl -qTHREAD -L MQ_INSTALLATION_PATH/lib \
-lmqmcb_r -qLIB -ICOBCPY_VALUE
```

32bitová klientská aplikace s podporou podprocesů

```
$ 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
```

64bitová klientská aplikace bez podprocesů

```
$ cob2 -o amq0put0 amq0put0.cbl -q64 -L MQ_INSTALLATION_PATH/lib - lmqicb \
-qLIB -ICOBCPY_VALUE
```

64bitová serverová aplikace s podporou podprocesů

```
$ cob2_r -o amq0put0 amq0put0.cbl -q64 -qTHREAD -L MQ_INSTALLATION_PATH/lib \
-lmqmcb_r -qLIB -ICOBCPY_VALUE
```

64bitová klientská aplikace s podporou podprocesů

```
$ cob2_r -o amq0put0 amq0put0.cbl -q64 -qTHREAD -L MQ_INSTALLATION_PATH/lib \
-lmqicb_r -qLIB -ICOBCPY_VALUE
```

Příprava programů v jazyce COBOL pomocí funkce Micro Focus COBOL

Před kompilací programu nastavte proměnné prostředí následujícím způsobem:

```
export COBCPY=COBCPY_VALUE
export LIBPATH=MQ_INSTALLATION_PATH/lib:$LIBPATH
```

Chcete-li kompilovat 32bitový program COBOL pomocí funkce Micro Focus COBOL, zadejte:

- Server pro jazyk COBOL

```
$ cob32 -xvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb
```

- Klient pro COBOL

```
$ cob32 -xvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb
```

- Server s podporou podprocesů 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 funkce Micro Focus COBOL, zadejte:

- Server pro jazyk COBOL

```
$ cob64 -xvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcb
```

- Klient pro COBOL

```
$ cob64 -xvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicb
```

- Server s podporou podprocesů 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, naleznete v dokumentaci Micro Focus COBOL.

AIX *Příprava CICS aplikačních programů v AIX*

Tyto informace použijte při přípravě programů CICS v adresáři AIX.

Použijte moduly *přepínače XA* k propojení CICS s IBM MQ. Další informace o struktuře přepínače XA naleznete v tématu Struktury přepínačů XA.

K dispozici je ukázkový soubor se zdrojovým kódem, který vám umožňuje vyvíjet přepínače XA pro jiné zprávy transakcí. Název poskytnutého zaváděcího modulu přepínače je uveden v seznamuTabulka 145 na stránce 971.

Tabulka 145. Základní kód pro aplikační programy CICS v systému AIX: Inicializační rutina XA

Popis	**C (zdroj)**	**C (exec)-přidat do svého XAD.Stanza**
Inicializační rutina XA	amqzscix.c	*amqzsc* - CICS pro AIX

Použijte předem sestavenou verzi IBM MQ souboru načtení přepínače *amqzsc*, který je poskytován s produktem.

Vždy propojte transakce C s knihovnou IBM MQ zajišťující neporušenost vláken *libmqm_r.a.*, a vaše transakce v jazyce COBOL s knihovnou COBOL *libmqmcb_r.a.*.

Další informace o podpoře transakcí CICS naleznete v příručce Administrace IBM MQ IBM MQ System Administration Guide.

AIX *Podpora produktu TXSeriesCICS*

IBM MQ v systému AIX podporuje TXSeries CICS pomocí rozhraní XA. Ujistěte se, že jsou aplikace CICS propojeny s verzí knihoven IBM MQ s podprocesy.

Programy CICS můžete spustit pomocí IBM COBOL Set for AIX nebo Micro Focus COBOL. Následující sekce popisují rozdíl mezi spuštěnými programy CICS v IBM sadě COBOL pro AIX a v mikrofokusu COBOL.

Napište programy IBM MQ , které jsou načteny do stejné oblasti CICS v jazyce C nebo COBOL. Nelze vytvořit kombinaci volání jazyka C a COBOL MQI do stejné oblasti CICS . Většina volání MQI ve druhém použitém jazyce selže s kódem příčiny `MQRC_HOBJ_ERROR`.

Příprava programů CICS COBOL pomocí IBM sady COBOL pro AIX

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

Chcete-li použít IBM COBOL, postupujte takto:

1. Exportujte 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:

   ```
   cicstcl -l IBMCOB yourprog.ccp
   ```

Příprava programů CICS COBOL pomocí Micro Focus COBOL

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

Chcete-li použít Micro Focus COBOL, postupujte takto:

1. Přidejte modul běhové knihovny IBM MQ COBOL do běhové knihovny 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 vaše existující aplikace mají taková volání, doporučuje se tyto funkce přesunout z aplikací v jazyce COBOL do své vlastní knihovny, například `myMQ.so`. Po přesunutí funkcí nezahrnujte IBM MQ knihovnu `libmqmcbrt.o` při sestavování aplikace v jazyce COBOL pro CICS.

   Kromě toho, pokud vaše aplikace v jazyce COBOL neprovádí žádné volání MQI v jazyce COBOL, nepropojujte `libmqmz_r` s `cicsmkcobol`.

   Tím se vytvoří soubor jazykové metody Micro Focus COBOL a umožní knihovně CICS runtime COBOL volat systémy IBM MQ for AIX or Linux .

   Poznámka: Příkaz `cicsmkcobol` spusťte pouze v případě, že instalujete jeden z následujících produktů:

   - Nová verze nebo vydání Micro Focus COBOL
   - Nová verze nebo vydání produktu CICS for AIX
   - Nová verze nebo vydání jakéhokoli podporovaného databázového produktu (pouze pro transakce v jazyce COBOL)
   - Nová verze nebo vydání produktu IBM MQ

2. Exportujte následující proměnnou prostředí:

```
COBCPY= MQ_INSTALLATION_PATH/inc export COBCPY
```

3. Přeložte, zkompilujte a propojte program zadáním:

```
cicstcl -l COBOL -e yourprog.ccp
```

Příprava programů CICS C

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

Sestavte programy v jazyce CICS C pomocí standardních zařízení CICS :

1. Exportujte ***jednu*** 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:

```
cicstcl -l C amqscic0.ccs
```

CICS ukázková transakce C

Ukázkový zdroj C pro transakci AIX IBM MQ poskytuje AMQSCIC0.CCS. Transakce čte zprávy z přenosové fronty SYSTEM.SAMPLE.CICS.WORKQUEUE ve výchozím správci front a umístí je do lokální fronty s názvem fronty obsaženým v záhlaví přenosu zprávy. Všechna selhání jsou odeslána do fronty SYSTEM.SAMPLE.CICS.DLQ. Použijte ukázkový skript MQSC AMQSCIC0.TST pro vytvoření těchto front a ukázkových vstupních front.

IBM i Sestavení vaší procedurální aplikace na IBM i

Publikace IBM i popisují, jak sestavit spustitelné aplikace z programů, které píšete, ke spuštění s produktem IBM i na systémech iSeries nebo System i .

Toto téma popisuje další úlohy a změny standardních úloh, které musíte provést při sestavování procedurálních aplikací IBM MQ for IBM i ke spuštění na systémech IBM i . Jsou podporovány programovací jazyky COBOL, C, C + +, Java a RPG. Informace o přípravě programů C++ viz Použití C++. Informace o přípravě programů Java naleznete v části Použití IBM MQ classes for Java.

Úlohy, které musíte provést, abyste vytvořili spustitelnou aplikaci IBM MQ for IBM i , závisí na programovacím jazyku, ve kterém je napsán zdrojový kód. Kromě kódování volání MQI ve zdrojovém kódu musíte přidat příslušné jazykové příkazy, které budou obsahovat soubory definice dat IBM MQ for IBM i pro jazyk, který používáte. Seznamte se s obsahem těchto souborů. Úplný popis naleznete v části “Soubory definice dat IBM MQ” na stránce 696 .

IBM i *Příprava programů v jazyce C v adresáři IBM i*

Produkt IBM MQ for IBM i podporuje zprávy o velikosti až 100 MB. Aplikační programy napsané v jazyce ILE C, které podporují IBM MQ zprávy větší než 16 MB, musí použít volbu kompilátoru `Teraspace` k přidělení dostatečné paměti pro tyto zprávy.

Další informace o volbách kompilátoru C naleznete v příručce *WebSphere Development Studio ILE C/C++ Programmer's Guide*.

Chcete-li zkompilovat modul C, můžete použít IBM i příkaz **CRTCMOD**. Při kompilaci se ujistěte, že knihovna obsahující soubory začlenění (QMQM) je v seznamu knihoven.

Poté musíte svázat výstup kompilátoru se servisním programem pomocí příkazu **CRTPGM** .

Tabulka 146. Příklad příkazu CRTPGM v prostředích bez podprocesů a podprocesů		
Typ prostředí	**Příkaz**	**Typ programu/ukončení**

Prostředí bez podprocesů	`CRTPGM PGM( pgmname )`
	`MODULE( pgmname )`
	`BNDSRVPGM(QMQM/LIBMQM)`
		Server nebo klient pro C

Prostředí s podporou podprocesů	`CRTPGM PGM( pgmname )`
	`MODULE( pgmname )`
	`BNDSRVPGM(QMQM/LIBMQM_R)`
		Server nebo klient pro C

kde *pgmname* je název vašeho programu.

Tabulka 147 na stránce 974 uvádí seznam knihoven, které jsou potřebné při přípravě programů v jazyce C na systému IBM i v prostředí bez podprocesů a v prostředí s podporou podprocesů.

Tabulka 147. Knihovny potřebné pro prostředí bez podprocesů a s podporou podprocesů		
Typ prostředí	**Soubor knihovny**	**Typ programu/ukončení**
Prostředí bez podprocesů	LIBMQM	Server pro C
	LIBMQIC & LIBMQM	Klient pro C
Prostředí s podporou podprocesů	LIBMQM_R	Server pro C
	LIBMQIC_R & LIBMQM_R	Klient pro C

IBM i *Příprava programů v jazyce COBOL v adresáři IBM i*

Seznamte se s přípravou programů v jazyce COBOL v produktu IBM i a s metodou 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 procedurální rozhraní volání poskytované servisními programy. To poskytuje přístup ke všem funkcím MQI v produktu IBM MQ for IBM i a podporu pro aplikace s podporou podprocesů. Toto rozhraní lze použít pouze s kompilátorem ILE COBOL.

Pro přístup k funkcím MQI se používá standardní syntaxe COBOL CALL.

Soubory kopie v jazyce COBOL obsahující pojmenované konstanty a definice struktury pro použití s rozhraním MQI jsou obsaženy ve zdrojovém fyzickém souboru QMQM/QCBLLESRC.

Kopírované soubory COBOL používají znak apostrofu (') jako oddělovač řetězců. Kompilátory jazyka COBOL IBM i předpokládají, že oddělovačem je uvozovka ("). Chcete-li zabránit kompilátoru generovat varovné zprávy, uveďte `OPTION (*APOST)` v příkazech **CRTCBLPGM**, **CRTBNDCBL**nebo **CRTCBLMOD**.

Chcete-li zajistit, aby kompilátor přijímal znak apostrofu (') jako oddělovač řetězců v kopírovaných souborech COBOL, použijte volbu kompilátoru \APOST.

Poznámka: Rozhraní dynamického volání není poskytováno v produktu IBM MQ 9.0 nebo novějším.

Chcete-li použít rozhraní volání vázané procedury, postupujte takto.

Postup

1. Vytvořte modul pomocí kompilátoru **CRTCBLMOD** , který určuje parametr:

```
LINKLIT(*PRC)
```

2. Použijte příkaz **CRTPGM** k vytvoření programového objektu a uveďte odpovídající parametr:

   Pro nezávitové aplikace:

```
BNDSRVPGM(QMQM/AMQ0STUB)        Server for COBOL for non-threaded applications
BNDSRVPGM(QMQM/AMQCSTUB)        Client for COBOL for non-threaded applications
```

   Pro závitové aplikace:

```
BNDSRVPGM(QMQM/AMQ0STUB_R)        Server for COBOL for threaded applications
BNDSRVPGM(QMQM/AMQCSTUB_R)        Client for COBOL for threaded applications
```

   Poznámka: Kromě programů vytvořených pomocí kompilátoru V4R4 ILE COBOL a obsahujících volbu THREAD (SERIALIZE) v příkazu PROCESS nesmí programy COBOL používat knihovny IBM MQ s podprocesy. I když byl program v jazyce COBOL tímto způsobem zabezpečen, buďte při návrhu aplikace opatrní, protože THREAD (SERIALIZE) vynucuje serializaci procedur jazyka COBOL na úrovni modulu a může ovlivnit celkový výkon.

   Další informace naleznete v příručce *WebSphere Development Studio: ILE COBOL Programmer's Guide* a v příručce *WebSphere Development Studio: ILE COBOL Reference* .

   Další informace o kompilaci aplikace CICS naleznete v příručce *CICS for IBM i Application Programming Guide*, SC41-5454.

IBM i *Příprava programů CICS v adresáři IBM i*

Seznamte se s kroky, které jsou vyžadovány při přípravě programů CICS v adresáři IBM i.

Chcete-li vytvořit program obsahující příkazy EXEC CICS a volání MQI, postupujte takto:

1. V případě potřeby 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 v jazyce COBOL.

   Zahrňte `CICSOPT(*NOGEN)` do příkazu CRTCICSC nebo CRTCICSCBL. Tím se zastaví zpracování, abyste mohli zahrnout příslušné servisní programy CICS a IBM MQ . Tento příkaz standardně vkládá kód do QTEMP/QACYCICS.
3. Kompilujte zdrojový kód pomocí příkazu CRTCMOD (pro program v jazyce C) nebo příkazu CRTCBLMOD (pro program v jazyce COBOL).
4. Pomocí příkazu CRTPGM propojte kompilovaný kód s příslušnými servisními programy CICS a IBM MQ . Tím se vytvoří spustitelný program.

Následuje příklad takového kódu (kompiluje 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 v jazyce RPG v adresáři IBM i*

Pokud používáte IBM MQ for IBM i, můžete psát své aplikace v RPG.

Další informace naleznete v části “Kódování IBM MQ programů v RPG (pouzeIBM i)” na stránce 1022a v příručce IBM i Application Programming Reference (ILE/RPG).

IBM i *Aspekty programování SQL pro IBM i*

Získejte informace o krocích, které jsou nezbytné při sestavování aplikace na systému IBM i pomocí SQL.

Pokud váš program obsahuje příkazy EXEC SQL a volání MQI, postupujte takto:

1. Přeložte příkazy EXEC SQL do příkazů nativního jazyka. Použijte příkaz CRTSQLCI pro program C. Použijte příkaz CRTSQLCBLI pro program v jazyce COBOL.

   Zahrnout `OPTION(*NOGEN)` do příkazu CRTSQLCI nebo CRTSQLCBLI. Tím se zastaví zpracování, abyste mohli zahrnout příslušné servisní programy IBM MQ . Tento příkaz standardně vkládá kód do QTEMP/QSQLTEMP.
2. Kompilujte zdrojový kód pomocí příkazu CRTCMOD (pro program v jazyce C) nebo příkazu CRTCBLMOD (pro program v jazyce COBOL).
3. Použijte CRTPGM k propojení kompilovaného kódu s příslušnými servisními programy IBM MQ . Tím se vytvoří spustitelný program.

Následuje příklad takového kódu (kompiluje 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)
```

Linux Sestavení vaší procedurální aplikace na Linux

Tyto informace popisují další úlohy a změny standardních úloh, které musíte provést při sestavování IBM MQ pro Linux aplikace ke spuštění.

Jsou podporovány jazyky C a C + +. Informace o přípravě programů C++ viz Použití C++.

Linux *Příprava programů v jazyce C v adresáři Linux*

Předkompilované programy v jazyce C 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 naleznete v tématu Standardy kódování na 64bitových platformách.

Propojování knihoven

V následujících tabulkách jsou uvedeny knihovny, které jsou potřebné při přípravě programů v jazyce C v systému Linux.

- Své programy musíte propojit s příslušnou knihovnou poskytovanou společností IBM MQ.

  V prostředí bez podprocesů vytvořte odkaz pouze na jednu z následujících knihoven:

Soubor knihovny	Typ programu/ukončení
libmqm.so	Server pro C
libmqic.so & libmqm.so	Klient pro C

  V prostředí s podporou podprocesů vytvořte 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. Nelze vytvořit odkaz na více než jednu knihovnu. To znamená, že nemůžete současně odkazovat na knihovnu s podporou podprocesů i na knihovnu bez podpory podprocesů.
2. Pokud píšete instalovatelnou službu (další informace viz Administrace IBM MQ), musíte se odkázat na knihovnu `libmqmzf.so` .
3. Pokud produkujete aplikaci pro externí koordinaci pomocí správce transakcí podporujícího standard XA, jako např. IBM TXSeries Encina nebo BEA Tuxedo, musíte se odkázat na `libmqmxa.so` (nebo `libmqmxa64.so` , pokud váš správce transakcí považuje typ 'long' za 64bitový) a knihovny `libmqz.so` v aplikaci bez podprocesů a na `libmqmxa_r.so` (nebo `libmqmxa64_r.so`). a knihovny `libmqz_r.so` v aplikaci s podporou podprocesů.
4. Knihovny IBM MQ musíte propojit před ostatními knihovnami produktu.

Linux *Vytváření 31bitových aplikací*

Toto téma obsahuje příklady příkazů použitých k sestavení 31bitových programů v různých prostředích.

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

Klientská aplikace v jazyce C, 31bitová, 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 v jazyce C, 31bitová, podprocesová

```
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 v jazyce C, 31bitová, 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 v jazyce C, 31bitová, 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++, 31bitová, 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
```

Aplikace klienta C++, 31bitová, podprocesová

```
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++, 31bitová, nepodprocesová

```
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++, 31bitová, s podporou podprocesů

```
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
```

Uživatelská procedura klienta jazyka C, 31bitová, bez podprocesů

```
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
```

Uživatelská procedura klienta C, 31bitová, podprocesová

```
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
```

Uživatelská procedura serveru C, 31bitová, 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, 31bitové, 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
```

Linux *Sestavení 32bitových aplikací*

Toto téma obsahuje příklady příkazů používaných k sestavení 32bitových programů v různých prostředích.

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

Klientská aplikace v jazyce 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 v jazyce C, 32bitová, podprocesová

```
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 v jazyce 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 v jazyce C, 32bitová, s podporou podprocesů

```
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, s podporou podprocesů

```
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 v jazyce C++, 32bitová, bez podpory 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 v jazyce C++, 32bitová, s podporou podprocesů

```
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
```

Uživatelská procedura klienta jazyka C, 32bitová, bez podprocesů

```
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
```

Uživatelská procedura klienta C, 32bitová, s podporou podprocesů

```
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
```

Ukončení serveru C, 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
```

Ukončení serveru C, 32bitový, s podporou podprocesů

```
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
```

Linux *Sestavení 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 v jazyce 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
```

Klientská aplikace v jazyce C, 64bitová, s podporou podprocesů

```
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 v jazyce 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
```

Serverová aplikace v jazyce C, 64bitová, s podporou podprocesů

```
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
```

Aplikace klienta C++, 64bitová, podprocesová

```
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, s podporou podprocesů

```
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
```

Uživatelská procedura klienta jazyka C, 64bitová, 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
```

Uživatelská procedura klienta C, 64bitová, s podporou podprocesů

```
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
```

Ukončení serveru C, 64bitový, 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
```

Ukončení serveru C, 64bitový, s podporou podprocesů

```
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 v jazyce COBOL v adresáři Linux*

Seznamte se s přípravou programů v jazyce COBOL v systému Linux a přípravou programů v jazyce COBOL pomocí IBM COBOL pro Linux na systémech x86 a Micro Focus COBOL.

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

1. 32bitové knihy pro kopírování v jazyce COBOL jsou nainstalovány v následujícím adresáři:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy32
   ```

   a symbolické odkazy jsou vytvořeny v:

   ```
   MQ_INSTALLATION_PATH/inc
   ```

2. Na 64bitových platformách se 64bitové kopírovací knihy COBOL instalují do následujícího adresáře:

   ```
   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 jedním z následujících:

Soubor knihovny	Typ programu/ukončení
libmqmcb.so	Server pro jazyk COBOL
libmqicb.so	Klient pro COBOL
libmqmcb_r.so	Server pro COBOL (aplikace s podporou podprocesů)
libmqicb_r.so	Klient pro COBOL (aplikace s podporou podprocesů)

Příprava programů v jazyce COBOL pomocí produktu IBM COBOL for Linux on x86

Ukázkové programy COBOL jsou dodávány s produktem IBM MQ. Chcete-li takový program zkompilovat, 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
```

32bitová klientská aplikace 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
```

32bitová klientská 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 -lmqicb_r -ICOBCPY_VALUE
```

Příprava programů v jazyce COBOL pomocí funkce 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 v jazyce COBOL, kde je podporován, pomocí funkce 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 funkce 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, naleznete v dokumentaci Micro Focus COBOL.

Windows Sestavení vaší procedurální aplikace na Windows

Příručky systémů Windows popisují, jak sestavit spustitelné aplikace z programů, které píšete.

Toto téma popisuje další úlohy a změny ve standardních úlohách, které musíte provést při sestavování aplikací IBM MQ for Windows ke spuštění v systémech Windows . Jsou podporovány programovací jazyky C, C + +, COBOL a Visual Basic. Informace o přípravě programů C++ viz Použití C++.

Úlohy, které musíte provést, chcete-li vytvořit spustitelnou aplikaci pomocí produktu IBM MQ for Windows , se liší podle programovacího jazyka, ve kterém je napsán váš zdrojový kód. Kromě kódování volání MQI ve zdrojovém kódu musíte přidat příslušné jazykové příkazy, aby byly zahrnuty soubory začlenění IBM MQ for Windows pro jazyk, který používáte. Seznamte se s obsahem těchto souborů. Úplný popis naleznete v části “Soubory definice dat IBM MQ” na stránce 696 .

Windows *Sestavení 64bitových aplikací na systému Windows*

V systému IBM MQ for Windowsjsou podporovány 32bitové i 64bitové aplikace. Spustitelné soubory a soubory knihovny IBM MQ jsou dodávány ve 32bitovém i 64bitovém formátu. Použijte příslušnou verzi v závislosti na aplikaci, se kterou pracujete.

Spustitelné soubory a knihovny

32bitová i 64bitová verze knihoven IBM MQ jsou dodávány v následujících umístěních:

Tabulka 148. 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 normální práci i po migraci. 32bitové 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 tak, aby hledala ve složce obsahující 32bitové knihovny.

Windows *Příprava programů v jazyce C v adresáři Windows*

Pracujte ve svém typickém Windows prostředí; IBM MQ for Windows nevyžaduje nic zvláštního.

Další informace o programování 64bitových aplikací viz Standardy kódování na 64bitových platformách.

- Propojte své programy s příslušnými knihovnami poskytovanými společností IBM MQ:

Soubor knihovny	Typ programu/ukončení
MQ_INSTALLATION_PATH \Tools\Lib\mqm.lib	server pro 32bitové C
MQ_INSTALLATION_PATH \Tools\Lib\mqic.lib	klient pro 32bitové prostředí C
MQ_INSTALLATION_PATH \Tools\Lib\mqicxa.lib	klient pro 32-bit C s koordinací transakcí
MQ_INSTALLATION_PATH \Tools\Lib64\mqm.lib	server pro 64bitové prostředí C
MQ_INSTALLATION_PATH \Tools\Lib64\mqic.lib	klient pro 64bitové prostředí C
MQ_INSTALLATION_PATH \Tools\Lib64\mqicxa.lib	klient pro 64-bit C s koordinací transakcí

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

Následující příkaz uvádí příklad kompilace ukázkového programu amqsget0 (pomocí kompilátoru Microsoft Visual C++).

Pro 32bitové 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 píšete instalovatelnou službu (další informace viz Administrace IBM MQ), musíte se odkázat na knihovnu mqmzf.lib .
- Pokud produkujete aplikaci pro externí koordinaci pomocí správce transakcí vyhovujícího standardu XA, jako např. IBM TXSeries Encina nebo BEA Tuxedo, musíte se odkázat na knihovnu mqmxa.lib nebo mqmxa.lib .
- Pokud píšete uživatelskou proceduru CICS , odkaz na knihovnu mqmcics4.lib .
- Knihovny IBM MQ musíte propojit před ostatními knihovnami produktu.

- Knihovny DLL musí být v cestě (PATH), kterou jste zadali.
- Pokud použijete malá písmena, kdykoli je to možné, můžete se přesunout z produktu IBM MQ for Windows na systémy IBM MQ for AIX or Linux , kde je použití malých písmen nezbytné.

Příprava programů CICS a Transaction Server

Ukázkový zdroj C pro transakci CICS IBM MQ poskytuje AMQSCIC0.CCS. Sestavíte jej pomocí standardních zařízení CICS . Například pro TXSeries pro Windows 2000:

1. Nastavte proměnnou prostředí (zadejte na jeden řádek 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. Přeložte, zkompilujte a propojte ukázkový 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 transakcí CICS naleznete v souboru Administrace IBM MQ.

Windows Příprava programů v jazyce COBOL v adresáři Windows

Pomocí těchto informací se naučíte připravovat programy v jazyce COBOL v adresáři Windowsa připravovat programy v jazycích CICS a Transaction Server.

1. 32bitové kopírovací knihy COBOL jsou nainstalovány v následujícím adresáři: *MQ_INSTALLATION_PATH* \Tools\cobol\CopyBook.
2. 64bitové knihy kopií v jazyce COBOL jsou nainstalovány 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 v systémech Windows , propojte program s jednou z následujících knihoven, které poskytuje produkt IBM MQ:

Soubor knihovny	**Typ programu nebo ukončení**
MQ_INSTALLATION_PATH \Tools\Lib\mqmcb	32bitový server pro Micro Focus COBOL
MQ_INSTALLATION_PATH \Tools\Lib\mqiccb	32bitový klient pro Micro Focus COBOL
MQ_INSTALLATION_PATH \Tools\Lib64\mqmcb	64bitový server pro Micro Focus COBOL
MQ_INSTALLATION_PATH \Tools\Lib64\mqiccb	64bitový klient pro Micro Focus COBOL

Když 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 funkce Micro Focus COBOL

Znovu propojte všechny existující 32bitové programy IBM MQ Micro Focus COBOL používající buď `mqmcb.lib` , nebo `mqiccb.lib`, spíše než knihovny `mqmcbb` a `mqiccbb` .

Chcete-li například zkompilovat ukázkový program `amq0put0`pomocí funkce Micro Focus COBOL:

1. Nastavte proměnnou prostředí COBCPY tak, aby ukazovala na zakladače jazyka IBM MQ COBOL (zadejte na jeden řádek následující kód):

   ```
   set COBCPY= MQ_INSTALLATION_PATH\
   Tools\Cobol\Copybook
   ```

2. Zkompilujte program tak, aby vám poskytl objektový soubor:

   ```
   cobol amq0put0 LITLINK
   ```

3. Propojte soubor objektu s běhovém systémem.
   - 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 objektu pro použití na klientovi IBM MQ :

     ```
     cbllink amq0put0.obj mqiccb.lib
     ```

Příprava programů CICS a Transaction Server

Chcete-li zkompilovat a propojit program TXSeries for Windows NT, V5.1 pomocí jazyka IBM VisualAge COBOL:

1. Nastavte proměnnou prostředí (zadejte na jeden řádek 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. Přeložte, zkompilujte a propojte svůj program:

```
cicstcl -l IBMCOB myprog.ccp
```

To je popsáno v příručce *Transaction Server for Windows NT, V4 Application Programming Guide* .

Chcete-li zkompilovat a propojit program CICS pro 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 Visual Basic v adresáři Windows*

Informace, které je třeba zvážit při používání programů Microsoft Visual Basic na systému Windows.

Deprecated V produktu IBM MQ 9.0je podpora pro Microsoft Visual Basic 6.0 zamítnuta. Třídy IBM MQ pro .NET jsou doporučenými náhradními technologiemi. Další informace viz téma Vývoj aplikací .NET.

Poznámka: 64bitové verze souborů modulu Visual Basic nejsou dodány.

Chcete-li připravit programy Visual Basic na systému Windows, postupujte takto:

1. Vytvořit nový projekt
2. Přidejte dodaný soubor modulu CMQB.BAS, do projektu.
3. Přidejte další dodané soubory modulu, pokud je potřebujete:
   - CMQBB.BAS:
   - CMQCFB.BAS:
   - CMQXB.BAS:
   - CMQPSB.BAS: Publikovat/odebírat

Informace o použití volání MQCONNXAny z adresáře Visual Basicnaleznete v části “Kódování v souboru Visual Basic” na stránce 1018 .

Před provedením volání MQI v kódu projektu volejte proceduru MQ_SETDEFAULTS. Tato procedura nastaví výchozí struktury, které volání MQI vyžadují.

Určete, zda vytváříte server nebo klienta IBM MQ před kompilací nebo spuštěním projektu, nastavením proměnné podmíněné kompilace *MqType*. Nastavte *MqType* v projektu Visual Basic na 1 pro server nebo 2 pro klienta takto:

1. Vyberte nabídku Projekt.
2. Vyberte volbu *Name* Vlastnosti (kde *Name* je název aktuálního projektu).
3. V dialogovém okně vyberte kartu Make.
4. Do pole Argumenty podmíněné kompilace zadejte pro server toto:

```
MqType=1
```

nebo toto pro klienta:

```
MqType=2
```

Související pojmy

“Kódování v souboru Visual Basic” na stránce 1018
Informace, které je třeba vzít v úvahu při kódování programů IBM MQ v Microsoft Visual Basic. Parametr Visual Basic je podporován pouze v systému Windows.

Související odkazy

“Propojení aplikací Visual Basic s kódem IBM MQ MQI client” na stránce 888
Aplikace Microsoft Visual Basic můžete propojit s kódem IBM MQ MQI client na systému Windows.

Windows *Uživatelská procedura zabezpečení rozhraní 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 uživatelské procedury kanálu, který poskytuje ověřování pro kanály IBM MQ pomocí rozhraní SSPI (Security Services Programming Interface). Rozhraní SSPI poskytuje integrované prostředky zabezpečení systémů Windows .

Balíky zabezpečení jsou načteny buď z knihovny security.dll , nebo z knihovny secur32.dll. Tyto knihovny DLL jsou dodávány s vaším operačním systémem.

Jednosměrné ověření se poskytuje pomocí ověřovacích služeb NTLM. Obousměrné ověření se poskytuje pomocí ověřovacích služeb Kerberos .

Uživatelský program zabezpečení je dodáván ve formátu zdroje a objektu. Můžete použít objektový kód tak, jak je, nebo můžete použít zdrojový kód jako výchozí bod pro vytvoření vlastních uživatelských programů.

Další informace najdete v tématu “Použití uživatelské procedury zabezpečení rozhraní SSPI v systému Windows” na stránce 1096.

Úvod do uživatelských procedur 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 zahájí zabezpečené připojení, tj. první program, který získá řízení po vytvoření relace MCA, je znám jako *iniciátor kontextu*. Partnerský program je znám jako *kontextový příjemce*.

Následující tabulka zobrazuje některé typy kanálů, které jsou iniciátory kontextu, a jejich přidružené příjemce kontextu.

Tabulka 149. Iniciátory kontextu a jejich přidružené příjemce kontextu	
Iniciátor kontextu	**Kontextový akceptátor**
MQCHT_CLNTCONN	MQCHT_SVRCONN
MQCHT_RECEIVER	MQCHT_SENDER-odesilatel
MQCHT_CLUSRCVR	MQCHT_CLUSSDR

Uživatelský program zabezpečení má dva vstupní body:

- **SCY_NTLM**

  Používá ověřovací služby NTLM, které poskytují jednosměrné ověření. NTLM umožňuje serverům ověřovat identity svých klientů. Neumožňuje klientům ověřit identitu serveru nebo jeden server ověřit identitu jiného serveru. Ověření NTLM bylo navrženo pro síťové prostředí, ve kterém jsou servery považovány za skutečné.

- **SCY_KERBEROS**

  Používá služby vzájemného ověřování 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 dělají programy uživatelské procedury kanálu SSPI.

Dodané programy channel-exit poskytují buď jednosměrné, nebo dvousměrné (vzájemné) ověření partnerského systému při zavedení relace. Pro konkrétní kanál má každý uživatelský program přidruženého *činitele* (podobně jako ID uživatele, viz “IBM MQ řízení přístupu a Windows činitelé” na stránce 988). Spojení mezi dvěma uživatelskými programy je přidružení mezi těmito dvěma činitelé.

Po vytvoření základní relace se vytvoří zabezpečené připojení mezi dvěma programy uživatelské procedury zabezpečení (jeden pro odesílající MCA a druhý pro přijímající MCA). Posloupnost operací je následující:

1. Každý program je přidružen ke konkrétnímu činiteli, například jako výsledek operace explicitního přihlášení.
2. Iniciátor kontextu požaduje zabezpečené připojení s partnerem z balíku zabezpečení (pro Kerberos, pojmenovaného partnera) a obdrží token (nazvaný token1). Token je odeslán partnerskému programu pomocí již zavedené základní relace.
3. Partnerský program (služba Context Acceptor) předá token1 do balíku zabezpečení, který ověřuje, zda je iniciátor kontextu autentický. Pro NTLM je nyní navázáno připojení.
4. Pro uživatelskou proceduru zabezpečení Kerberos(tj. pro vzájemné ověření) vygeneruje balík zabezpečení také druhý token (nazvaný token2), který služba Context Acceptor vrátí iniciátoru kontextu pomocí základní relace.
5. Iniciátor kontextu používá token2 k ověření, že je služba Context Acceptor autentická.
6. Jsou-li v této fázi obě aplikace spokojeny s autenticitou tokenu partnera, vytvoří se zabezpečené (ověřené) připojení.

IBM MQ řízení přístupu a Windows činitelé

Řízení přístupu, které produkt IBM MQ poskytuje, je založeno na uživateli a skupině. Ověření, které produkt Windows poskytuje, je založeno na činitelích, jako např. uživatel a název služby

servicePrincipal(SPN). V případě názvu servicePrincipalmůže být mnoho z nich přidruženo k jednomu uživateli.

Uživatelská procedura zabezpečení SSPI používá pro ověření příslušné činitele Windows . Pokud je ověření Windows úspěšné, uživatelská procedura předá ID uživatele, které je přidruženo k Windows činiteli IBM MQ pro řízení přístupu.

Činitelé Windows , kteří jsou důležití pro ověření, se liší v závislosti na typu použitého ověření.

- V případě ověřování NTLM je činitel Windows pro iniciátor kontextu ID uživatele přidružené ke spuštěného procesu. Vzhledem k tomu, že toto ověření je jednosměrné, činitel přidružený ke službě Context Acceptor je irelevantní.
- Pro ověření Kerberos je činitel Windows na kanálech CLNTCONN ID uživatele přidružené ke spuštěného procesu. Jinak je činitel Windows názvem servicePrincipal, který je vytvořen přidáním následující předpony do názvu QueueManager.

```
ibmMQSeries/
```

z/OS Sestavení vaší 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 musíte provést při sestavování aplikací IBM MQ for z/OS pro tato prostředí. Jsou podporovány programovací jazyky COBOL, C, C + +, Assembler a PL/I. (Informace o sestavování aplikací C++ viz Použití C++.)

Úlohy, které musíte provést pro vytvoření spustitelné aplikace IBM MQ for z/OS , závisí jak na programovacím jazyku, ve kterém je program napsán, tak na prostředí, ve kterém bude aplikace spuštěna.

Kromě kódování volání MQI ve vašem programu přidejte příslušné příkazy jazyka, které budou obsahovat soubor definice dat IBM MQ for z/OS pro jazyk, který používáte. Seznamte se s obsahem těchto souborů. Úplný popis naleznete v části “Soubory definice dat IBM MQ” na stránce 696 .

Poznámka

Název **thlqual** je kvalifikátor vyšší úrovně instalační knihovny v systému z/OS.

z/OS *Příprava programu ke spuštění*

Poté, co jste napsali program pro aplikaci IBM MQ , abyste vytvořili spustitelnou aplikaci, musíte ji zkompilovat nebo sestavit a poté propojit výsledný kód objektu s programem stub, který produkt IBM MQ for z/OS dodává pro každé prostředí, které podporuje.

Příprava programu závisí na prostředí (dávka, CICS, IMS(BMP nebo MPP), Linux nebo z/OS UNIX System Services), ve kterém je aplikace spuštěna, a na struktuře datových sad ve vaší instalaci produktu z/OS .

“Dynamické volání stubu IBM MQ” na stránce 995 popisuje alternativní metodu provádění volání MQI ve vašich programech, abyste nemuseli vytvářet odkazy na úpravy stubu IBM MQ . Tato metoda není k dispozici pro všechny jazyky a prostředí.

Neodkazujte-neupravujte vyšší úroveň programu stub, než je verze produktu IBM MQ for z/OS , na které je váš program spuštěn. Například program spuštěný na systému MQSeries for OS/390, V5.2 nesmí být propojen s programem stub dodávaným s produktem IBM MQ for z/OS V7.

z/OS *Vytváření 64bitových aplikací v jazyce C*

V produktu z/OSjsou 64bitové aplikace v jazyce C sestaveny pomocí voleb kompilátoru a vázacího programu LP64 . Soubor záhlaví IBM MQ for z/OS *cmqc.h* rozpozná, kdy je tato volba poskytnuta kompilátoru, a vygeneruje datové typy a struktury IBM MQ vhodné pro 64bitovou operaci.

Kód C sestavený pomocí této volby musí být sestaven tak, aby používal knihovny DLL (dynamic-link libraries) odpovídající požadované sémantice koordinace. Chcete-li toho dosáhnout, svažte kompilovaný kód s příslušným postranním balíčkem definovaným v následující tabulce:

Tabulka 150. Pro každou sémantickou koordinaci je vyžadován název postranní plochy.

coordination	**Název postranní paluby**
Rozhraní MQI s jednofázovým potvrzením	CSQBMQ2X
Dvoufázové potvrzování s koordinací RRS pomocí sloves RRS	CSQBRR2X
Dvoufázové potvrzování s koordinací RRS pomocí příkazových slov MQI	CSQBRI2X

Poznámka: Pro 31bitové aplikace v jazyce C také nastavíte volby kompilátoru pro volající rozhraní (buď jazykové prostředí, nebo XPLINK), jak je popsáno v tématu “Sestavení dávkových aplikací z/OS pomocí 31bitového jazykového prostředí nebo XPLINK” na stránce 992. Pro 64bitové aplikace v jazyce C nezadáváte volající rozhraní, protože jediné podporované propojení je XPLINK.

Pomocí procedury EDCQCB JCL, dodávané s produktem *z/OS XL C/C++*, sestavte program IBM MQ pro jednofázové potvrzování jako dávkovou úlohu, jak je uvedeno níže:

```
//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 z/OS*

Naučte se, jak sestavit dávkové aplikace z/OS a kroky, které je třeba při tom zvážit.

Chcete-li sestavit aplikaci pro systém IBM MQ for z/OS , který bude spuštěn v dávce z/OS , vytvořte jazyk JCL (job control language), který bude provádět tyto úlohy:

1. Kompilujte (nebo sestavte) program pro vytvoření objektového kódu. JCL pro vaši kompilaci musí obsahovat příkazy SYSLIB, které zpřístupňují soubory definice dat produktu kompilátoru. Definice dat jsou dodávány v následujících knihovnách IBM MQ for z/OS :
   - Pro COBOL, **thlqual**.SCSQCOBC
   - V případě jazyka asembleru **thlqual**.SCSQMACS
   - Pro C, **thlqual**.SCSQC370
   - Pro PL/I **thlqual**.SCSQPLIC
2. V případě aplikace v jazyce C předem propojte kód objektu vytvořený v kroku “1” na stránce 990.
3. Pro aplikace PL/I použijte volbu kompilátoru EXTRN (SHORT).
4. Odkaz-upravte kód objektu vytvořený v kroku “1” na stránce 990 (nebo kroku “2” na stránce 990 pro aplikaci C), abyste vytvořili zaváděcí modul. Když odkazujete na úpravu kódu, musíte zahrnout

jeden z programů dávkového stubu IBM MQ for z/OS (CSQBSTUB nebo jeden z programů stubu RRS: CSQBRRSI nebo CSQBRSTB).

CSQBSTUB
jednofázové potvrzování poskytované IBM MQ for z/OS

CSQBRRSI
dvoufázové potvrzování poskytované RRS pomocí MQI

CSQBRSTB
dvoufázové potvrzování poskytované přímo RRS

Notes:

a. Pokud používáte CSQBRSTB, musíte také propojit-upravit aplikaci pomocí ATRSCSS z SYS1.CSSLIB. Obrázek 113 na stránce 991 a Obrázek 114 na stránce 991 k tomu zobrazují fragmenty JCL. Stuby jsou nezávislé na jazyku a jsou dodávány v knihovně **thlqual**.SCSQLOAD.

b. Pokud je vaše aplikace spuštěna pod jazykovým prostředím, měli byste se ujistit, že jste místo toho propojili úpravy s knihovnou DLL jazykového prostředí, jak je popsáno v tématu “Sestavení dávkových aplikací z/OS pomocí 31bitového jazykového prostředí nebo XPLINK” na stránce 992.

5. Uložte zaváděcí modul do zaváděcí knihovny aplikace.

```
⋮
//*
//* WEBSPHERE MQ FOR Z/OS LIBRARY CONTAINING BATCH STUB
//*
//CSQSTUB   DD  DSN=++THLQUAL++.SCSQLOAD,DISP=SHR
//*
⋮
//SYSIN     DD *
  INCLUDE CSQSTUB(CSQBSTUB)
⋮
/*
```

Obrázek 113. Fragmenty JCL pro odkazování-úpravy modulu objektu v dávkovém prostředí pomocí jednofázového potvrzování

```
⋮
//*
//* 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 114. Fragmenty JCL pro propojení-úpravy modulu objektu v dávkovém prostředí pomocí dvoufázového potvrzování

Chcete-li spustit dávku nebo program RRS, 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 do knihovny STEPLIB používané relací TSO.

Chcete-li spustit dávkový program ze shellu z/OS UNIX System Services , přidejte knihovny **thlqual**.SCSQAUTH a **thlqual**.SCSQLOAD do specifikace STEPLIB ve vašem profilu $HOME? .profile takto:

```
STEPLIB= thlqual.SCSQAUTH: thlqual.SCSQLOAD
export STEPLIB
```

z/OS *Sestavení dávkových aplikací z/OS pomocí 31bitového jazykového prostředí nebo XPLINK*

Produkt IBM MQ for z/OS poskytuje sadu knihoven DLL (dynamic link libraries), které se musí použít při úpravě odkazů na aplikace.

Existují dvě varianty knihoven, které umožňují aplikaci používat jedno z následujících volajících rozhraní:

- Rozhraní volání 31bitového jazykového prostředí.
- 31bitové rozhraní volání XPLINK. z/OS XPLINK je konvence volání s vysokým výkonem, která je k dispozici pro aplikace v jazyce C. Viz XPLINK | NOXPLINK v dokumentaci k systému z/OS 2.2 .

Chcete-li používat knihovny DLL, je aplikace svázána nebo propojena s tzv. *sidedecks*, namísto stubů poskytovaných s dřívějšími verzemi. Sidedecks se nacházejí v knihovně SCSQDEFS (místo knihovny SCSQLOAD).

Tabulka 151. Varianty knihoven dynamických odkazů

Potvrdit	**Knihovna DLL 31bitového jazykového prostředí**	**31bitová knihovna DLL XPLINK**	**Ekvivalentní název stubu**
1 knihovny MQI s fázovým potvrzením	CSQBMQ1	CSQBMQ1X	CSQBSTUB
Dvoufázové potvrzování s koordinací RRS pomocí řídicích sloves transakcí RRS	CSQBRR1	CSQBRR1X	CSQBRSTB
Dvoufázové potvrzování s koordinací RRS pomocí řídicích sloves transakcí MQI	CSQBRI1	CSQBRI1X	CSQBRRSI

Poznámka: Všechny sidedecks obsahují definici vstupního bodu převodu dat MQXCNVC, který byl dříve vyřešen zahrnutím CSQASTUB.

Společné otázky:

- Následující zpráva se zobrazí v protokolu úlohy, pokud vaše aplikace používá asynchronní spotřebu zpráv (volání MQCB, MQCTL nebo MQSUB) a předchozí rozhraní DLL není použito:

  ```
  CSQB001E Programy jazykového prostředí spuštěné v dávce z/OS nebo z/OS UNIX System Services
  musí používat rozhraní DLL k IBM MQ
  ```

  Řešení: Znovu sestavte aplikaci pomocí sidedecks namísto stubů, jak bylo podrobně popsáno dříve.
- V době sestavení programu se zobrazí následující zpráva

  ```
  IEW2469E Atributy odkazu na MQAPI-NAME ze sekce your-code se neshodují s atributy
  cílový symbol
  ```

  Příčina: To znamená, že jste kompilovali program XPLINK s verzí produktu V701 (nebo novější) cmqc.h, ale nejste vázáni s sidedecks.

  Řešení: Změňte soubor sestavení programu tak, aby byl svázán s příslušným sidedeckem z SCSQDEFS namísto stubu z SCSQLOAD.

Následující ukázka JCL demonstruje, jak můžete kompilovat a linkovat program C tak, aby používal 31 bitové volající rozhraní 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** . Linkování používá volbu **DYNAM=DLL** a odkazuje na knihovnu **CSQBMQ1** .

Následující ukázka JCL demonstruje, jak můžete kompilovat a linkovat program v jazyce C tak, aby používal 31 bitové volající rozhraní knihovny DLL XPLINK:

```
//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** . Linkování používá volbu **DYNAM=DLL** a odkazuje na knihovnu **CSQBMQ1X** .

Ujistěte se, že jste přidali knihovnu DLL volby kompilace do každého programu v modulu. Zprávy jako IEW2456E 9207 SYMBOL CSQ1BAK NEVYŘEŠENÉ indikují, že je třeba zkontrolovat, zda byly všechny programy kompilovány s volbou DLL.

z/OS *Sestavení aplikací CICS v produktu z/OS*

Tyto informace použijte při sestavování aplikací CICS v produktu z/OS.

Chcete-li sestavit aplikaci pro systém IBM MQ for z/OS , který je spuštěn v adresáři CICS, musíte:

- Přeložte příkazy CICS ve vašem programu do jazyka, ve kterém je napsán zbytek vašeho programu.
- Kompilujte nebo sestavte výstup z překladače pro vytvoření objektového kódu.
  - Pro programy PL/I použijte volbu kompilátoru EXTRN (SHORT).
  - V případě aplikací v jazyce C, pokud aplikace nepoužívá XPLINK, použijte volbu kompilátoru DEFINE (MQ_OS_LINKAGE=1).
- Odkaz-upravte kód objektu a vytvořte zaváděcí modul.

Produkt CICS poskytuje proceduru pro provedení těchto kroků v posloupnosti pro každý programovací jazyk, který podporuje.

- V případě CICS Transaction Server for z/OSpopisuje příručka *CICS Transaction Server for z/OS System Definition Guide* , jak tyto procedury používat, a příručka *CICS/ESA Application Programming Guide* poskytuje další informace o procesu překladu.

Musíte zahrnout:

- V příkazu SYSLIB fáze kompilace (nebo sestavení) se jedná o příkazy, které zpřístupňují soubory definice dat produktu kompilátoru. Definice dat jsou dodávány v následujících knihovnách IBM MQ for z/OS :
  - Pro COBOL, **thlqual**.SCSQCOBC
  - V případě jazyka asembleru **thlqual**.SCSQMACS
  - Pro C, **thlqual**.SCSQC370
  - Pro PL/I **thlqual**.SCSQPLIC
- V JCL linkování se jedná o program stub IBM MQ for z/OS CICS (CSQCSTUB). Obrázek 115 na stránce 994 zobrazuje fragmenty kódu JCL, které to mají provést. Stub je nezávislý na jazyku a je dodáván 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 115. Fragmenty JCL pro propojení-úpravy modulu objektu v prostředí CICS

- V případě verzí produktu CICS novějších než CICS TS 3.2nebo, chcete-li použít rozhraní API vlastností zpráv IBM MQ nebo rozhraní IBM MQ API MQCB, MQCTL, MQSTAT, MQSUB nebo MQSUBR, musíte propojit kód objektu se stubem dodaným CICS , DFHMQSTB, a nikoli IBM MQ dodaným CSQCSTUB. Další informace o sestavování programů IBM MQ pro systém CICSnaleznete v tématu Program stub rozhraní API pro přístup k IBM MQ voláním MQI v dokumentaci k produktu CICS .

Po provedení těchto kroků uložte zaváděcí modul do zaváděcí knihovny aplikace a definujte program pro CICS obvyklým způsobem.

Před spuštěním programu CICS musí administrátor systému definovat program CICS jako program a transakci IBM MQ . Poté jej můžete spustit obvyklým způsobem.

z/OS *Sestavení aplikací IMS (BMP nebo MPP)*

Tyto informace použijte při sestavování aplikací IMS (BMP nebo MPP).

Pokud sestavujete dávkové programy DL/I, viz “Sestavení dávkových aplikací z/OS” na stránce 990. Chcete-li sestavit další aplikace, které jsou spuštěny v adresáři IMS (buď jako BMP, nebo MPP), vytvořte soubor JCL, který bude provádět tyto úlohy:

1. Kompilujte (nebo sestavte) program pro vytvoření objektového kódu. JCL pro vaši kompilaci musí obsahovat příkazy SYSLIB, které zpřístupňují soubory definice dat produktu kompilátoru. Definice dat jsou dodávány v následujících knihovnách IBM MQ for z/OS :
   - Pro COBOL, **thlqual**.SCSQCOBC
   - Pro jazyk asembleru, **thlqual**.SCSQMACS
   - Pro C, **thlqual**.SCSQC370
   - Pro PL/I, **thlqual**.SCSQPLIC
2. V případě aplikace v jazyce C předem propojte modul objektu vytvořený v kroku “1” na stránce 994.
3. Pro programy PL/I použijte volbu kompilátoru EXTRN (SHORT).
4. V případě aplikace v jazyce C, pokud aplikace nepoužívá XPLINK, použijte volbu kompilátoru DEFINE (MQ_OS_LINKAGE=1).
5. Odkaz-upravte kód objektu vytvořený v kroku “1” na stránce 994 (nebo krok “2” na stránce 994 pro aplikaci C/370), abyste vytvořili zaváděcí modul:
   a. Zahrňte modul jazykového rozhraní IMS (DFSLI000).

b. Zahrňte program stub IBM MQ for z/OS IMS (CSQQSTUB). Obrázek 116 na stránce 995 ukazuje fragmenty JCL, které to mají provést. Stub je nezávislý na jazyku a je dodáván v knihovně **thlqual**.SCSQLOAD.

   Poznámka: Používáte-li jazyk COBOL, vyberte volbu kompilátoru NODYNAM, abyste povolili editoru sestavení interpretovat odkazy na CSQQSTUB, pokud nemáte v úmyslu použít dynamické propojení, jak je popsáno v tématu "Dynamické volání stubu IBM MQ" na stránce 995.

6. Uložte zaváděcí modul do zaváděcí knihovny aplikace.

```
⋮
//*
//* WEBSPHERE MQ FOR Z/OS LIBRARY CONTAINING IMS STUB
//*
//CSQSTUB   DD  DSN=thlqual.SCSQLOAD,DISP=SHR
//*
⋮
//LKED.SYSIN DD *
  INCLUDE CSQSTUB(CSQQSTUB)
  ⋮
/*
```

Obrázek 116. Fragmenty JCL pro propojení-úpravy modulu objektu v prostředí IMS

Před spuštěním programu IMS musí administrátor systému definovat program IMS jako program a transakci IBM MQ : poté jej můžete spustit obvyklým způsobem.

z/OS *Sestavování aplikací produktu z/OS UNIX System Services*

Tyto informace použijte při sestavování aplikací z/OS UNIX System Services .

Chcete-li sestavit aplikaci v jazyce C pro systém IBM MQ for z/OS , který je spuštěn v adresáři z/OS UNIX System Services, zkompilujte a propojte aplikaci následujícím způsobem:

```
cc -o mqsamp -W c,DLL -I "//' thlqual.SCSQC370'" mqsamp.c "//' thlqual.SCSQDEFS(CSQBMQ1)'"
```

kde **thlqual** je kvalifikátor vyšší úrovně používaný vaší instalací.

Chcete-li spustit program v jazyce C, musíte do svého souboru `.profile` přidat následující položky. To by mělo být ve vašem kořenovém adresáři:

```
STEPLIB= thlqual.SCSQANLE:thlqual.SCSQAUTH: STEPLIB
```

Všimněte si, že musíte ukončit z/OS UNIX System Servicesa znovu zadat z/OS UNIX System Services , aby byla změna rozpoznána.

Chcete-li spustit více shellů, přidejte na začátek řádku slovo export, tj.:

```
export STEPLIB= thlqual.SCSQANLE:thlqual.SCSQAUTH: STEPLIB
```

Po úspěšném dokončení můžete propojit volání CSQBSTUB a vydat volání IBM MQ .

"Dynamické volání stubu IBM MQ" na stránce 995 popisuje alternativní metodu provádění volání MQI ve vašich programech, abyste nemuseli vytvářet odkazy na úpravy stubu IBM MQ . Tato metoda není k dispozici pro všechny jazyky a prostředí.

Neodkazujte-neupravujte vyšší úroveň programu stub, než je verze produktu IBM MQ for z/OS , na které je váš program spuštěn. Například program spuštěný na systému IBM WebSphere MQ for z/OS 7.1 nesmí být propojen s programem typu stub dodávaným s produktem IBM MQ for z/OS 8.0.

z/OS *Dynamické volání stubu IBM MQ*

Místo odkazování na program stub IBM MQ s kódem objektu můžete dynamicky volat stub z vašeho programu.

To můžete provést v dávkovém prostředí, v prostředí IMSa v prostředí CICS . Toto zařízení není v prostředí RRS podporováno. Pokud váš aplikační program používá RRS ke koordinaci aktualizací, viz “Aspekty RRS” na stránce 1000.

Tato metoda však:

- Zvyšuje složitost vašich programů.
- Zvyšuje paměť vyžadovanou vašimi programy v době provádění.
- Snižuje výkon vašich programů.
- Znamená, že nemůžete používat stejné programy v jiných prostředích

Pokud stub voláte dynamicky, musí být v době provádění k dispozici příslušný program stub a jeho aliasy. Chcete-li to zajistit, zahrňte datovou sadu IBM MQ for z/OS SCSQLOAD:

- V případě dávky a IMSve zřetězení STEPLIB JCL.
- Pro parametr CICSve zřetězení CICS DFHRPL.

V případě systému IMSse ujistěte, že knihovna obsahující dynamický stub (sestavená podle popisu v informacích o instalaci adaptéru IMS v části Nastavení adaptéru IMS) je před datovou sadou SCSQLOAD ve zřetězení STEPLIB kódu JCL oblasti.

Při dynamickém volání stubu použijte názvy zobrazené v souboru Tabulka 152 na stránce 996 . V PL/I deklarujte pouze názvy volání použité ve vašem programu.

Tabulka 152. Názvy volání pro dynamické propojení

Volání rozhraní MQI	Názvy dynamických volání dávky (jiné než RRS)	CICS názvy dynamických volání	IMS názvy dynamických volání
MQBACK	CSQBBACK	není podporováno	Nepodporováno
MQBUFMH	CSQBBFMH	CSQCBFMH [1]	MQBUFMH
MQCB	CSQBCB	CSQCCB [1]	Nepodporováno
MQCLOSE	CSQBCLOS	CSQCCLOS	MQCLOSE
MQCMIT	CSQBCOMM	není podporováno	Nepodporováno
MQCONN	CSQBCONN	CSQCCONN	MQCONN
MQCONNX	CSQBCONX	CSQCCONX	MQCONNX
MQCRTMH	CSQBCTMH	CSQCCTMH [1]	MQCRTMH
MQCTL	CSQBCTL	CSQCCTL [1]	Nepodporováno
MQDISC	CSQBDISC	CSQCDISC	MQDISC
MQDLTMH	CSQBDTMH	CSQCDTMH [1]	MQDLTMH
MQDLTMP	CSQBDTMP	CSQCDTMP [1]	MQDLTMP
MQGET	CSQBGET	CSQCGET	MQGET
MQINQ	CSQBINQ	CSQCINQ	MQINQ
MQINQMP	CSQBIQMP	CSQCIQMP [1]	MQINQMP
MQMHBUF	CSQBMHBF	CSQCMHBF [1]	MQMHBUF
MQOPEN	CSQBOPEN	CSQCOPEN	MQOPEN
MQPUT	CSQBPUT	CSQCPUT	MQPUT
MQPUT1	CSQBPUT1	CSQCPUT1	MQPUT1

Tabulka 152. Názvy volání pro dynamické propojení (pokračování)			
Volání rozhraní MQI	**Názvy dynamických volání dávky (jiné než RRS)**	**CICS názvy dynamických volání**	**IMS názvy dynamických volání**
MQSET	CSQBSET	CSQCSET	MQSET
MQSETMP	CSQBSTMP	CSQCSTMP [1]	MQSETMP
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 v případě, že používáte produkt CICS TS 3.2 nebo novější a musíte použít CSQCSTUB dodávaný s produktem CICS . Pro CICS TS 3.2musí být použita oprava APAR PK66866 . Pro CICS TS 4.1musí být použita oprava APAR PK89844 .

Příklady použití této techniky naleznete na následujících obrázcích:

- Dávka a COBOL: viz Obrázek 117 na stránce 997
- CICS a COBOL: viz Obrázek 118 na stránce 998
- IMS a COBOL: viz Obrázek 119 na stránce 998
- Dávkový a sestavovací modul: viz Obrázek 120 na stránce 998
- CICS a assembleru: viz Obrázek 121 na stránce 998
- IMS a assembleru: viz Obrázek 122 na stránce 999
- Dávka a C: Obrázek 123 na stránce 999
- CICS a C: viz Obrázek 124 na stránce 999
- IMS a C: viz Obrázek 125 na stránce 999
- Dávka a PL/I: viz Obrázek 126 na stránce 999
- IMS a PL/I: viz Obrázek 127 na stránce 1000

```
...
       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 117. 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 118. Dynamické propojování pomocí jazyka COBOL v prostředí 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 119. Dynamické propojování pomocí jazyka COBOL v prostředí IMS

```
...
        LOAD   EP=CSQBOPEN
...
        CALL (15),(HCONN,MQOD,OPTIONS,HOBJ,COMPCODE,REASON),VL
...
        DELETE EP=CSQBOPEN
...
```

Obrázek 120. Dynamické propojování pomocí 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 121. Dynamické propojování pomocí jazyka sestavení v prostředí CICS

```
...
         LOAD   EP=MQOPEN
...
         CALL (15),(HCONN,MQOD,OPTIONS,HOBJ,COMPCODE,REASON),VL
...
         DELETE EP=MQOPEN
...
```

Obrázek 122. Dynamické propojování pomocí jazyka sestavení v prostředí 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 123. Dynamické propojování pomocí jazyka C v dávkovém prostředí

```
...
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 124. 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 125. 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 126. Dynamické propojení pomocí 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 127. Dynamické propojení pomocí PL/I v prostředí IMS

z/OS *Aspekty RRS*

Zvažte použití těchto informací, pokud váš aplikační program používá RRS ke koordinaci aktualizací.

Produkt IBM MQ poskytuje dva různé stuby pro dávkové programy, které vyžadují koordinaci RRS- viz “Dávkový adaptér RRS” na stránce 861. Rozdíl v chování pozdějších volání rozhraní API je určen v čase MQCONN dávkovým adaptérem z informací předaných rutinou stubu v rozhraní API MQCONN nebo MQCONNX. To znamená, že dynamická volání rozhraní API jsou k dispozici pro dávkové programy, které vyžadují koordinaci RRS, za předpokladu, že počáteční připojení k produktu IBM MQ bylo provedeno pomocí příslušného stubu. Toto ilustruje následující příklad:

```
       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.
```

z/OS *Ladění programů*

Pomocí těchto informací získáte informace o ladění programů TSO a CICS a přehled o trasování CICS .

Hlavními pomůckami pro ladění aplikačních programů IBM MQ for z/OS jsou kódy příčiny vrácené každým voláním rozhraní API. Seznam těchto opatření, včetně nápadů na nápravná opatření, viz:

- položky IBM MQ for z/OS zprávy, dokončení, a kódy příčiny pro IBM MQ for z/OS
- Zprávy a kódy příčiny pro všechny ostatní IBM MQ platformy

Toto téma také navrhuje další ladicí nástroje pro použití v konkrétních prostředích.

Ladění programů TSO

Pro programy TSO jsou k dispozici následující interaktivní ladicí nástroje:

- Nástroj TEST
- Interaktivní ladicí nástroj VS COBOL II
- Interaktivní ladicí nástroj INSPECT pro programy C a PL/I

Ladění programů CICS

Pomocí nástroje CICS Execution Diagnostic Facility (CEDF) můžete interaktivně testovat své programy CICS , aniž byste museli upravovat program nebo proceduru přípravy programu.

Další informace o EDF naleznete v příručce *CICS Transaction Server for z/OS CICS Application Programming Guide*.

CICS trasování

Pravděpodobně bude také užitečné použít k řízení aktivity trasování CICS transakci CETR (CICS Trace Control transaction).

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 číslo trasování.

Chcete-li interpretovat položky trasování CICS , viz Tabulka 153 na stránce 1001.

Položka trasování CICS pro tyto hodnoty je AP0*xxx* (kde *xxx* je číslo trasování určené při povolení adaptéru CICS). Všechny trasovací záznamy kromě CSQCTEST jsou vydány CSQCTRUE. CSQCTEST vydávají CSQCRST a CSQCDSP.

Tabulka 153. Položky trasování adaptéru CICS

Název	Popis	Posloupnost trasování	Trasovat data
CSQCABNT	Nestandardní ukončení	Před zadáním příkazu END_THREAD ABNORMÁLNÍ do souboru IBM MQ. Důvodem je konec úlohy a implicitní vrácení může být provedeno aplikací. Požadavek ROLLBACK je v tomto případě zahrnut do volání END_THREAD.	Informace o jednotce práce. Tyto informace můžete použít při zjišťování stavu práce. (Lze jej například ověřit proti výstupu vytvořenému příkazem DISPLAY THREAD nebo obslužným programem pro tisk protokolu IBM MQ for z/OS .)
CSQCBACK	Vrácení synchronizačního bodu	Před zadáním příkazu BACKOUT do adresáře IBM MQ for z/OS. Důvodem je explicitní požadavek na vrácení z aplikace.	Informace o jednotce práce.
CSQCCCRC	Kód dokončení a kód příčiny	Po neúspěšném návratu z volání rozhraní API.	Kód dokončení a kód příčiny.
CSQCCOMM	Potvrzení synchronizačního bodu	Před vydáním příkazu COMMIT pro IBM MQ for z/OS. To může být způsobeno požadavkem na jednofázové potvrzení nebo druhou fází požadavku na dvoufázové potvrzení. Požadavek je způsoben explicitním požadavkem na synchronizační bod z aplikace.	Informace o jednotce práce.

Tabulka 153. Položky trasování adaptéru CICS (pokračování)

Název	Popis	Posloupnost trasování	Trasovat data
CSQCEXER	Provedení vyřešení	Před vydáním příkazu EXECUTE_RESOLVE na IBM MQ for z/OS.	Informace o pracovní jednotce pracovní jednotky vydávající příkaz EXECUTE_RESOLVE. Toto je poslední neověřená jednotka práce v procesu resynchronizace.
CSQCGETW	Čekání GET	Před vydáním příkazu CICS počkejte.	Adresa ECB, na kterou se má čekat.
CSQCGMGD	Data zprávy GET	Po úspěšném návratu z MQGET.	Až 40 bajtů dat zprávy.
CSQCGMGH	Manipulátor zprávy GET	Před zadáním příkazu MQGET do souboru IBM MQ for z/OS.	Popisovač objektu.
CSQCGMGI	Získat ID zprávy	Po úspěšném návratu z MQGET.	ID zprávy a ID korelace zprávy.
CSQCINDL	Seznam nejistých	Po úspěšném návratu z druhého INQUIRE_INDOUBT.	Seznam neověřených jednotek práce.
CSQCINDO	IBM používat pouze		
CSQCINDS	Velikost seznamu Nejistých	Po úspěšném návratu z prvního seznamu INQUIRE_INDOUBT a seznamu neověřených není prázdný.	Délka seznamu. Děleno 64 udává počet neověřených jednotek práce.
CSQCINQH	Manipulátor INQ	Před zadáním MQINQ do adresáře IBM MQ for z/OS.	Popisovač objektu.
CSQCLOSH	Popisovač CLOSE	Před zadáním příkazu MQCLOSE do adresáře IBM MQ for z/OS.	Popisovač objektu.
CSQCLOST	Odebrání ztraceno	Během procesu resynchronizace produkt CICS informuje adaptér, že byl restartován, takže nejsou k dispozici žádné informace o odebrání týkající se jednotky práce, která je resynchronizována.	ID pracovní jednotky známé produktu CICS pro jednotku práce, která se znovu synchronizuje.
CSQCNIND-nastavení	Dispozice není nejistá	Během procesu resynchronizace produkt CICS informuje adaptér, že jednotka práce, která se resynchronizuje, by neměla být neověřená (to znamená, že je možná stále spuštěná).	ID pracovní jednotky známé produktu CICS pro jednotku práce, která se znovu synchronizuje.
CSQCNORT	Normální ukončení	Před zadáním příkazu END_THREAD NORMAL do souboru IBM MQ for z/OS. Důvodem je konec úlohy, a proto může aplikace provést implicitní potvrzení synchronizačního bodu. V tomto případě je do volání END_THREAD zahrnut požadavek COMMIT.	Informace o jednotce práce.

Tabulka 153. Položky trasování adaptéru CICS (pokračování)

Název	Popis	Posloupnost trasování	Trasovat data
CSQCOPNH	Manipulátor OPEN	Po úspěšném návratu z MQOPEN.	Popisovač objektu.
CSQCOPNO	objekt OPEN	Před zadáním příkazu MQOPEN do adresáře IBM MQ for z/OS.	Název objektu.
CSQCPMGD	Data zprávy PUT	Před zadáním příkazu MQPUT do souboru IBM MQ for z/OS.	Až 40 bajtů dat zprávy.
CSQCPMGH	Manipulátor zprávy PUT	Před zadáním příkazu MQPUT do souboru IBM MQ for z/OS.	Popisovač objektu.
CSQCPMGI	ID zprávy PUT	Po úspěšném provedení MQPUT z IBM MQ for z/OS.	ID zprávy a ID korelace zprávy.
CSQCPREP	Připravit synchronizační bod	Před vydáním příkazu PREPARE pro IBM MQ for z/OS v první fázi zpracování dvoufázového potvrzování. Toto volání lze také zadat z komponenty distribuovaného řazení do fronty jako volání rozhraní API.	Informace o jednotce práce.
CSQCP1MD	Data zprávy PUTONE	Před zadáním příkazu MQPUT1 do 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 zadáním příkazu MQPUT1 do IBM MQ for z/OS.	Název objektu.
CSQCRBAK	Vyřešené vrácení	Před vydáním příkazu RESOLVE_ROLLBACK pro IBM MQ for z/OS.	Informace o jednotce práce.
CSQCRCMT	Vyřešené potvrzení	Před zadáním příkazu RESOLVE_COMMIT do souboru IBM MQ for z/OS.	Informace o jednotce práce.
CSQCRMIR	Odezva vyvolání RMI	Před návratem do vyvolání CICS RMI (rozhraní správce prostředků) ze specifického vyvolání.	Hodnota odezvy RMI architektury. Jeho význam závisí na typu vyvolání. Tyto hodnoty jsou zdokumentovány v příručce *CICS Transaction Server for z/OS Customization Guide*. Chcete-li určit typ vyvolání, podívejte se na předchozí položky trasování vytvořené komponentou vyvolání RMI CICS .
CSQCRSYN	Resynchronizace	Před spuštěním procesu resynchronizace pro úlohu.	ID pracovní jednotky známé produktu CICS pro jednotku práce, která se znovu synchronizuje.
CSQCSETH	Manipulátor SET	Před zadáním příkazu MQSET do adresáře IBM MQ for z/OS.	Popisovač objektu.
CSQCTASE	IBM používat pouze		

Tabulka 153. Položky trasování adaptéru CICS (pokračování)			
Název	**Popis**	**Posloupnost trasování**	**Trasovat data**
CSQCTEST	Test trasování	Používá se ve volání EXEC CICS ENTER TRACE k ověření čísla trasování dodaného uživatelem nebo stavu trasování připojení.	Žádná data.
CSQCDCFF	IBM používat pouze		

Zpracování chyb procedurálních programů

Tyto informace vysvětlují chyby související s voláními rozhraní MQI aplikací, a to buď při volání, nebo při doručení zprávy do konečného místa určení.

Kdykoli je to možné, vrátí správce front všechny chyby ihned po volání MQI. Jedná se o *lokálně určené chyby*.

Při odesílání zpráv do vzdálené fronty nemusí být při volání MQI zřejmé chyby. V tomto případě správce front, který identifikuje chyby, je ohlásí odesláním další zprávy do původního programu. Jedná se o *vzdáleně určené chyby*.

Lokálně určené chyby

Informace o lokálně určených chybách, které zahrnují: selhání volání MQI, přerušení systému a zprávy obsahující nesprávná data.

Tři nejčastější příčiny chyb, které může správce front okamžitě nahlásit, jsou:

- Selhání volání MQI; například kvůli zaplnění fronty
- Přerušení běhu některé části systému, na které aplikace závisí; například správce front.
- Zprávy obsahující data, která nelze úspěšně zpracovat

Pokud používáte asynchronní prostředek vložení, chyby se okamžitě nehlásí. Pomocí volání MQSTAT načtěte informace o stavu předchozích asynchronních operací vložení.

Selhání volání MQI

Správce front může okamžitě hlásit jakékoli chyby v kódování volání MQI. Provádí to pomocí sady předdefinovaných návratových kódů. Ty jsou rozděleny do kódů dokončení a kódů příčiny.

Chcete-li zobrazit, zda bylo volání úspěšné, vrátí správce front po dokončení volání *kód dokončení* . Existují tři kódy dokončení, které označují úspěch, částečné dokončení a selhání volání. Správce front také vrací *kód příčiny* , který označuje příčinu částečného dokončení nebo selhání volání.

Kódy dokončení a příčiny pro každé volání jsou uvedeny s popisem tohoto volání v poli Návratový kód. Podrobnější informace, včetně nápadů na nápravná opatření, viz:

- z/OS položky IBM MQ for z/OS zprávy, dokončení, a kódy příčiny pro IBM MQ for z/OS
- Zprávy a kódy příčiny pro všechny ostatní IBM MQ platformy

Navrhněte své programy pro zpracování všech návratových kódů, které mohou vzniknout z každého volání.

System ipřerušení

Aplikace si nemusí být vědoma přerušení, pokud se správce front, ke kterému je připojena, musí zotavit ze selhání systému. Musíte však navrhnout aplikaci, abyste se ujistili, že vaše data nebudou ztracena, pokud dojde k takovému přerušení.

Metody, které můžete použít, abyste se ujistili, že vaše data zůstanou konzistentní, závisí na platformě, na které je spuštěn váš správce front:

z/OS

V prostředích CICS a IMS můžete provádět volání MQPUT a MQGET v rámci pracovních jednotek spravovaných produktem CICS nebo IMS. V dávkovém prostředí můžete provádět volání MQPUT a MQGET stejným způsobem, ale musíte deklarovat synchronizační body pomocí:

- volání IBM MQ for z/OS MQCMIT a MQBACK (viz “Potvrzení a zálohování jednotek práce” na stránce 825) nebo
- Služba z/OS Správa transakcí a obnovitelné služby Resource Manager (RRS) poskytuje podporu dvoufázových synchronizačních bodů. Služba RRS umožňuje aktualizovat prostředky produktu IBM MQ i další prostředky produktu s podporou služby RRS, například prostředky uložené procedury Db2 , v rámci jediné logické pracovní jednotky. Informace o podpoře synchronizačních bodů RRS viz “Správa transakcí a obnovitelné služby správce prostředků” na stránce 829.

IBM i

Volání MQPUT a MQGET můžete provádět v rámci globálních pracovních jednotek spravovaných vázaným zpracováním IBM i . Synchronizační body můžete deklarovat pomocí nativních příkazů IBM i COMMIT a ROLLBACK nebo pomocí příkazů specifických pro daný jazyk. Lokální pracovní jednotky jsou spravovány produktem IBM MQ pomocí volání MQCMIT a MQBACK.

Systémy AIX, Linux, and Windows

V těchto prostředích můžete provádět volání MQPUT a MQGET obvyklým způsobem, ale musíte deklarovat synchronizační body pomocí volání MQCMIT a MQBACK (viz “Potvrzení a zálohování jednotek práce” na stránce 825). V prostředí CICS jsou příkazy MQCMIT a MQBACK zakázány, protože můžete provádět volání MQPUT a MQGET v rámci pracovních jednotek spravovaných produktem CICS.

Použijte trvalé zprávy pro přenos všech dat, která si nemůžete dovolit ztratit. Trvalé zprávy jsou obnoveny ve frontách, pokud se správce front musí zotavit ze selhání. ALW Při IBM MQ zapnuté AIX, Linux, and Windowsse volání MQGET nebo MQPUT v rámci aplikace nezdaří v okamžiku zaplnění všech souborů protokolu zprávou MQRC_RESOURCE_PROBLEM. Další informace o souborech protokolu v systému AIX, Linux, and Windowsviz Administrace IBM MQ. z/OS z/OS viz Plánování v z/OS.

Pokud je správce front zastaven operátorem v době, kdy je spuštěna aplikace, obvykle se používá 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, rychlé aplikace mohou pravděpodobně ignorovat stav uvedení do klidového stavu a pokračovat, dokud se neukončí jako normální. Déle běžící aplikace nebo aplikace, které čekají na doručení zpráv, by měly při použití volání MQOPEN, MQPUT, MQPUT1a MQGET použít volbu *při uvedení do klidového stavu* . Tyto volby znamenají, že volání selžou při uvedení správce front do klidového stavu, ale aplikace může mít stále čas na čisté ukončení vyvoláním volání, která ignorují stav uvedení do klidového stavu. Takové aplikace mohou také potvrdit nebo vrátit zpět změny, které provedly, a poté ukončit.

Pokud je vynuceno zastavení správce front (tj. zastavení bez uvedení do klidového stavu), aplikace obdrží při volání MQI kód příčiny MQRC_CONNECTION_BROKEN. Ukončete aplikaci nebo případně v systémech IBM i IBM MQ for IBM i, AIX, Linux, and Windows zadejte volání MQDISC.

Zprávy obsahující nesprávná data

Používáte-li v aplikaci pracovní jednotky a program nemůže úspěšně zpracovat zprávu, kterou načte z fronty, je volání MQGET odvoláno.

Správce front udržuje počet (v poli *BackoutCount* deskriptoru zprávy), kolikrát se to stane. Udržuje tento počet v deskriptoru každé zprávy, která je ovlivněna. Tento počet může poskytnout cenné informace o efektivitě aplikace. Zprávy s počtem vrácení, které se v průběhu času zvyšují, jsou opakovaně odmítány; navrhněte svou aplikaci tak, aby analyzovala důvody a podle toho tyto zprávy obsluhuje.

z/OS Chcete-li v systému IBM MQ for z/OSzajistit, aby počet vrácení přežil restartování správce front, nastavte atribut **HardenGetBackout** na hodnotu MQQA_BACKOUT_HARDENED; jinak, pokud je nutné restartovat správce front, neudržuje přesný počet vrácení pro každou zprávu. Nastavení atributu tímto způsobem přidá pokutu za další zpracování.

V systémech IBM MQ for IBM i IBM i, AIX, Linux, and Windows počet vrácení vždy přežije restartování správce front.

z/OS Také v systému IBM MQ for z/OS, když odeberete zprávy z fronty v rámci pracovní jednotky, můžete označit jednu zprávu, aby nebyla znovu zpřístupněna, pokud je pracovní jednotka odvolána aplikací. S označenou zprávou se zachází, jako by byla načtena pod novou pracovní jednotkou. Zprávu, která má být vynechána, označíte pomocí volby MQGMO_MARK_SKIP_BACKOUT.(ve struktuře MQGMO) při použití volání MQGET. Další informace o této technice viz “Přeskočení vrácení” na stránce 773 .

Použití zpráv sestavy pro určování problémů

Vzdálený správce front nemůže hlásit chyby, například selhání při vložení zprávy do fronty při volání MQI, ale může vám odeslat zprávu sestavy, která informuje o tom, jak zprávu zpracovala.

V rámci aplikace můžete vytvářet zprávy sestav (MQPUT) a také vybrat volbu pro jejich příjem (v takovém případě jsou odesílány jinou aplikací nebo správcem front).

Vytváření zpráv sestavy

Zprávy sestavy umožňují aplikaci sdělit jiné aplikaci, že nemůže pracovat se zprávou, která byla odeslána.

Na počátku však musí být analyzováno pole *Report* , aby se zjistilo, zda aplikace, která odeslala zprávu, má zájem být informována o jakýchkoli problémech. Poté, co jste zjistili, že zpráva sestavy je povinná, musíte se rozhodnout:

- Zda chcete zahrnout celou původní zprávu, pouze prvních 100 bajtů dat, nebo žádnou z původních zpráv.
- Co dělat s původní zprávou. Můžete jej vyřadit nebo nechat přejít do fronty nedoručených zpráv.
- Zda je zapotřebí také obsah polí *MsgId* a *CorrelId* .

Použijte pole *Feedback* , abyste označili příčinu generování zprávy sestavy. Vložte zprávy sestavy do fronty odpovědí aplikace. Další informace viz Zpětná vazba .

Vyžádání a příjem zpráv sestavy (MQGET)

Když odešlete zprávu do jiné aplikace, nebudete informováni o žádných problémech, pokud nevyplníte pole *Report* a neoznačíte požadovanou zpětnou vazbu. Dostupné volby viz Struktura pole sestavy .

Správci front vždy vkládají zprávy sestav do fronty pro odpovědi aplikace a doporučuje se, aby vaše vlastní aplikace učinily totéž. Používáte-li prostředek zpráv sestavy, zadejte název fronty pro odpovědi v deskriptoru zpráv zprávy; jinak volání MQPUT selže.

Vaše aplikace musí obsahovat procedury, které monitorují vaši frontu pro odpověď, a zpracovat všechny zprávy, které do ní dorazí. Nezapomeňte, že zpráva sestavy může obsahovat všechny původní zprávy, prvních 100 bajtů původní zprávy nebo žádnou z původních zpráv.

Správce front nastaví pole *Feedback* zprávy sestavy tak, aby uváděli příčinu chyby; například cílová fronta neexistuje. Vaše programy by měly dělat totéž.

Další informace o zprávách sestavy viz “Zprávy sestavy” na stránce 20.

Vzdáleně určené chyby

Při odesílání zpráv do vzdálené fronty i v případě, že lokální správce front zpracoval volání MQI bez nalezení chyby, mohou další faktory ovlivnit způsob zpracování zprávy vzdáleným správcem front.

Například fronta, na kterou cílíte, může být plná nebo nemusí existovat. Pokud má být vaše zpráva zpracována jinými intermediační správci front na trase do cílové fronty, může některá z těchto zpráv najít chybu.

Problémy při doručování zprávy

Pokud volání MQPUT selže, můžete se pokusit znovu vložit zprávu do fronty, vrátit ji odesílateli nebo ji vložit do fronty nedoručených zpráv.

Každá volba má své opodstatnění, ale možná nebudete chtít znovu zkusit vložit zprávu, pokud příčinou nezdaru operace MQPUT byla skutečnost, že cílová fronta byla plná. V tomto případě jeho vložení do fronty nedoručených zpráv umožňuje jeho pozdější doručení do správné cílové fronty.

Opakovat doručení zprávy

Před vložením zprávy do fronty nedoručených zpráv se vzdálený správce front pokusí zprávu znovu vložit do fronty, pokud byly pro kanál nastaveny atributy *MsgRetryCount* a *MsgRetryInterval* nebo pokud existuje uživatelský program pro opakování pokusů, který má použít (jehož název je uveden v poli *MsgRetryExitId* atributu kanálu).

Je-li pole *MsgRetryExitId* prázdné, použijí se hodnoty v atributech *MsgRetryCount* a *MsgRetryInterval* .

Pokud pole *MsgRetryExitId* není prázdné, spustí se uživatelský program s tímto názvem. Další informace o použití vlastních uživatelských programů naleznete v části “Programy uživatelské procedury kanálu pro kanály systému zpráv” na stránce 928.

Vrátit zprávu odesílateli

Zprávu vrátíte odesílateli tak, že požádáte o vygenerování zprávy sestavy, která bude obsahovat všechny původní zprávy.

Podrobnosti o volbách zprávy sestavy viz “Zprávy sestavy” na stránce 20 .

Použití fronty nedoručených zpráv (nedoručených zpráv)

Nemůže-li správce front doručit zprávu, pokusí se zprávu vložit do fronty nedoručených zpráv. Tato fronta by měla být definována při instalaci správce front.

Vaše programy mohou používat frontu nedoručených zpráv stejným způsobem, jakým ji používá správce front. Název fronty nedoručených zpráv můžete najít otevřením objektu správce front (pomocí volání MQOPEN) a dotazem na atribut **DeadLetterQName** (pomocí volání MQINQ).

Když správce front vloží zprávu do této fronty, přidá do zprávy záhlaví, jehož formát je popsán strukturou záhlaví nedoručených zpráv (MQDLH); viz MQDLH-záhlaví nedoručených zpráv. Toto záhlaví zahrnuje název cílové fronty a důvod, proč byla zpráva vložena do fronty nedoručených zpráv. Musí být odebrán a problém musí být vyřešen před vložením zprávy do zamýšlené fronty. Správce front také změní pole *Format* deskriptoru zprávy (MQMD), aby označil, že zpráva obsahuje strukturu MQDLH.

Struktura MQDLH

Doporučuje se přidat strukturu MQDLH ke všem zprávám, které jste vložili do fronty nedoručených zpráv. Pokud však hodláte používat obslužnou rutinu nedoručených zpráv poskytovanou určitými produkty IBM MQ , musíte do zpráv přidat strukturu MQDLH.

Přidání záhlaví do zprávy může způsobit, že je zpráva pro frontu nedoručených zpráv příliš dlouhá, takže se vždy ujistěte, že vaše zprávy jsou kratší než maximální velikost povolená pro frontu nedoručených zpráv, alespoň o hodnotu konstanty MQ_MSG_HEADER_LENGTH. Maximální velikost zpráv povolených ve frontě je určena hodnotou atributu **MaxMsgLength** fronty. Pro frontu nedoručených zpráv se ujistěte, že je tento atribut nastaven na maximum povolené správcem front. Pokud vaše aplikace nemůže doručit zprávu a zpráva je příliš dlouhá na to, aby mohla být vložena do fronty nedoručených zpráv, postupujte podle pokynů uvedených v popisu struktury MQDLH.

Ujistěte se, že fronta nedoručených zpráv je monitorována a že všechny zprávy, které do ní dorazí, jsou zpracovány. Obslužná rutina fronty nedoručených zpráv je spuštěna jako dávkový obslužný program a lze ji použít k provádění různých akcí s vybranými zprávami ve frontě nedoručených zpráv. Další podrobnosti viz “Zpracování fronty nedoručených zpráv” na stránce 1008.

Je-li převod dat nezbytný, správce front převede informace záhlaví při použití volby MQGMO_CONVERT ve volání MQGET. Je-li proces vkládání zprávy MCA, za záhlavím následuje veškerý text původní zprávy.

Zprávy vložené do fronty nedoručených zpráv mohou být ořízнuty, pokud jsou pro tuto frontu příliš dlouhé. Možnou indikací této situace jsou zprávy ve frontě nedoručených zpráv, které mají stejnou délku jako hodnota atributu **MaxMsgLength** fronty.

Zpracování fronty nedoručených zpráv
Tyto informace obsahují obecné informace o programovacím rozhraní při použití zpracování fronty nedoručených zpráv.

Zpracování fronty nedoručených zpráv závisí na požadavcích lokálního systému, ale při vytváření specifikace zvažte následující skutečnosti:

- Zprávu lze identifikovat jako zprávu se záhlavím fronty nedoručených zpráv, protože hodnota pole formátu v deskriptoru MQMD je MQFMT_DEAD_LETTER_HEADER.
- Pokud při IBM MQ for z/OS použití CICSagent MCA vloží tuto zprávu do fronty nedoručených zpráv, pole *PutApplType* bude MQAT_CICS a pole *PutApplName* bude *ApplId* systému CICS následované názvem transakce agenta MCA.
- Příčina, proč má být zpráva směrována do fronty nedoručených zpráv, je obsažena v poli *Reason* záhlaví fronty nedoručených zpráv.
- Záhlaví fronty nedoručených zpráv obsahuje podrobnosti o názvu cílové fronty a názvu správce front.
- Záhlaví fronty nedoručených zpráv obsahuje pole, která musí být obnovena v deskriptoru zprávy před vložením zprávy do cílové fronty. Patří mezi ně:
  1. *Encoding*
  2. *CodedCharSetId*
  3. *Format*
- Deskriptor zprávy je stejný jako PUT původní aplikace, s výjimkou tří zobrazených polí (Encoding, CodedCharSetIda Format).

Aplikace fronty nedoručených zpráv musí provést jednu nebo více následujících akcí:

- Zkontrolujte pole *Reason* . Zpráva mohla být vložena agentem MCA z následujících důvodů:
  - Zpráva byla delší než maximální velikost zprávy pro kanál.

    Příčina je MQRC_MSG_TOO_BIG_FOR_CHANNEL
  - Zprávu nelze vložit do cílové fronty

    Příčinou je libovolný kód příčiny MQRC_ *, který může vrátit operace MQPUT .
  - Uživatelská procedura požadovala tuto akci

    Kód příčiny je ten, který byl dodán uživatelskou procedurou, nebo výchozí MQRC_SUPPRESSED_BY_EXIT
- Pokuste se zprávu předat do zamýšleného místa určení, kde je to možné.
- Uchovejte zprávu po určitou dobu před vyřazením, když je určena příčina odklonu, ale ne okamžitě opravitelná.
- Dejte administrátorům pokyny k nápravě problémů, kde byly určeny.
- Vyřadit zprávy, které jsou poškozené nebo jinak nezpracovatelné.

Existují dva způsoby, jak se vypořádat se zprávami, které jste obnovili z fronty nedoručených zpráv:

1. Pokud je zpráva určena pro lokální frontu:
   - Proveďte všechny překlady kódu potřebné k extrakci dat aplikace
   - Provést převody kódu na těchto datech, pokud se jedná o lokální funkci
   - Vložit výslednou zprávu do lokální fronty se všemi podrobnostmi o obnoveném deskriptoru zprávy
2. Pokud je zpráva určena pro vzdálenou frontu, vložte ji do fronty.

Informace o způsobu zpracování nedoručených zpráv v prostředí distribuovaných front naleznete v tématu Co se stane, když zprávu nelze doručit?.

Programování vícesměrového vysílání

Tyto informace slouží k získání informací o programovacích úlohách výběrového vysílání produktu IBM MQ , jako je například připojení ke správci front a vytváření sestav výjimek.

Výběrové vysílání produktu IBM MQ bylo navrženo tak, aby bylo pro uživatele co nejtransparentnější, a přesto bylo kompatibilní s existujícími aplikacemi. Definování objektu COMMINFO a nastavení parametrů **MCAST** a **COMMINFO** objektu TOPIC znamená, že existující aplikace IBM MQ nevyžadují podstatné přepsání pro použití výběrového vysílání. Mohou však existovat určitá omezení (další informace naleznete v části “Výběrové vysílání a rozhraní MQI” na stránce 1009) a některé problémy se zabezpečením, které je třeba zvážit (další informace naleznete v tématu Zabezpečení výběrového vysílání).

Výběrové vysílání a rozhraní MQI

Tyto informace slouží k pochopení hlavních koncepcí rozhraní MQI (Message Queue Interface) a jejich vztahu k produktu IBM MQ Multicast.

Odběry výběrového vysílání jsou přechodné. Protože nejsou zahrnuty žádné fyzické fronty, není možné ukládat zprávy offline vytvořené trvalými odběry.

Poté, co se aplikace přihlásí k odběru tématu výběrového vysílání, obdrží zpět popisovač objektu, ze kterého může přijímat nebo MQGET, jako by se jednalo o popisovač fronty. To znamená, že jsou podporovány pouze spravované odběry výběrového vysílání (odběry vytvořené pomocí příkazu MQSO_MANAGED), tj. není možné vytvořit odběr a 'nasměrování' zpráv do fronty. To znamená, že zprávy musí být spotřebovány z popisovače objektu vráceného při volání odběru. V klientu jsou zprávy uloženy ve vyrovnávací paměti zpráv, dokud nejsou spotřebovány klientem. Další informace naleznete v oddílu MessageBuffer konfiguračního souboru klienta . Pokud klient nedrží krok s rychlostí publikování, jsou zprávy podle potřeby vyřazeny a nejstarší zprávy jsou vyřazeny jako první.

Obvykle se jedná o administrační rozhodnutí, zda aplikace používá výběrové vysílání, či nikoli, určené nastavením atributu MCAST objektu TOPIC. Pokud publikační aplikace musí zajistit, aby výběrové vysílání nebylo použito, může použít volbu `MQOO_NO_MULTICAST` . Podobně může odebírající aplikace zajistit, aby výběrové vysílání nebylo používáno přihlášením k odběru s volbou `MQSO_NO_MULTICAST` .

Výběrové vysílání IBM MQ podporuje použití selektorů zpráv. Selektor je používán aplikací k registraci jejího zájmu pouze o zprávy s vlastnostmi, které splňují dotaz SQL92 reprezentovaný výběrovým řetězcem. Další informace o selektorech zpráv viz “Selektory.” na stránce 30.

V následující tabulce jsou uvedeny všechny hlavní koncepty MQI a jejich vztah k výběrové vysílání:

Tabulka 154. Koncepce MQI a jejich vztah k výběrovému vysílání

Koncepce rozhraní MQI	**Akce při pokusu o použití výběrového vysílání**	**Kód příčiny**
Vložení zprávy s nulovou délkou	Odmítnuto	2005 (07D5) (RC2005): MQRC_BUFFER_LENGTH_ERROR
Seskupení	Odmítnuto	2046 (07FE) (RC2046): MQRC_OPTIONS_ERROR
Segmentace	Odmítnuto	2443 (098B) (RC2443): MQRC_SEGMENTATION_NOT_ALLOWED
Distribuční seznamy	Odmítnuto	2154 (086A) (RC2154): MQRC_RECS_PRESENT_ERROR

Tabulka 154. Koncepce MQI a jejich vztah k výběrovému vysílání (pokračování)

Koncepce rozhraní MQI	Akce při pokusu o použití výběrového vysílání	Kód příčiny
MQINQ	Odmítnuto pro popisovače témat: MQINQ a MQSET témat není podporováno.	2038 (07F6) (RC2038): MQRC_NOT_OPEN_FOR_INQUIRE
MQINQ	Přijato pro spravovaný popisovač. Lze se dotazovat pouze na `Aktuální hloubku` .	• Je-li hodnota Aktuální hloubka, není k dispozici žádný použitelný kód příčiny. • Pokud je hodnota jiná než Aktuální hloubka, kód příčiny je 2067 (0813) (RC2067): MQRC_SELECTOR_ERROR.
MQSET	Odmítnuto pro všechny manipulátory.	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
Zamknout zprávy	Odmítnuto	2046 (07FE) (RC2046): MQRC_OPTIONS_ERROR
Procházet se značkou	Odmítnuto	2036 (07F4) (RC2036): MQRC_NOT_OPEN_FOR_BROWSE
Předat kontext	Odmítnuto	2046 (07FE) (RC2046): MQRC_OPTIONS_ERROR
MQPUT1	Zamítnuto. Pokus o zadání tématu MQPUT1 pouze pro výběrové vysílání je neplatný.	2560 (0A00) (RC2560): MQRC_MULTICAST_ONLY
Trvalý odběr	Odmítnuto, pokud je téma označeno jako "Pouze výběrové vysílání", jinak se provede odběr bez výběrového vysílání.	2436 (0984) (RC2436): MQRC_DURABILITY_NOT_ALLOWED
TopicString > 255	Zamítnuto. Je-li řetězec tématu delší než 255 znaků, bude v klientu odmítnut.	2425 (0979) (RC2425): MQRC_TOPIC_STRING_ERROR

Tabulka 154. Koncepce MQI a jejich vztah k výběrovému vysílání (pokračování)		
Koncepce rozhraní MQI	**Akce při pokusu o použití výběrového vysílání**	**Kód příčiny**
Byl proveden nespravovaný odběr	Odmítnuto, pokud je téma označeno jako "Pouze výběrové vysílání", jinak se provede odběr bez výběrového vysílání.	2046 (07FE) (RC2046): MQRC_OPTIONS_ERROR
MQPMO_NOT_OWN_SUBS	Odmítnuto	2046 (07FE) (RC2046): MQRC_OPTIONS_ERROR

Následující položky rozšiřují některé koncepty MQI z předchozí tabulky a poskytují informace o některých konceptech MQI, které nejsou v tabulce:

Trvalost zpráv
Pro přechodné odběratele výběrového vysílání jsou trvalé zprávy od vydavatele doručovány nezotavitelným způsobem.

Oříznutí zprávy
Oříznutí zpráv je podporováno, což znamená, že je možné, aby aplikace:

1. Zadejte příkaz MQGET.
2. Získání 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

Use this information to understand IBM MQ Multicast continuous peer-to-peer operation; although IBM MQ connects to an IBM MQ queue manager, messages do not flow through that queue manager.

Ačkoli musí být vytvořeno připojení ke správci front, aby mohl objekt tématu výběrového vysílání MQOPEN nebo MQSUB, samotné zprávy neprocházejí správcem front. Proto po dokončení operací MQOPEN nebo MQSUB v objektu tématu výběrového vysílání je možné pokračovat v přenosu zpráv výběrového vysílání i v případě, že došlo ke ztrátě připojení ke správci front. Existují dva způsoby provozu:

Ke správci front je vytvořeno normální připojení.
Komunikace výběrového vysílání je možná, když existuje připojení ke správci front. Pokud se připojení nezdaří, použijí se běžná pravidla MQI, například MQPUT na popisovač objektu výběrového vysílání vrátí hodnotu 2009 (07D9) (RC2009): MQRC_CONNECTION_BROKEN.

Ke správci front je vytvořeno připojení klienta k opětovnému připojení.
Multicast komunikace je možná i během cyklu opětovného připojení. To znamená, že i v případě přerušení připojení ke správci front nebude ovlivněno vkládání a konzumace zpráv výběrového vysílání. Klient se pokusí znovu připojit ke správci front a pokud se toto opětovné připojení nezdaří, dojde k přerušení manipulátoru připojení a k selhání všech volání MQI včetně volání výběrového vysílání. Další informace viz: Automatické opětovné připojení klienta

Pokud některá aplikace explicitně vydá příkaz MQDISC, budou všechny odběry výběrového vysílání a manipulátory objektů zavřeny.

Nepřetržitá operace typu peer-to-peer pro výběrové vysílání

Jednou z výhod komunikace mezi klienty na stejné úrovni je, že zprávy nemusí procházet správcem front. Pokud se tedy připojení ke správci front přeruší, přenos zpráv pokračuje. Pro požadavky na souvislé zprávy v tomto režimu platí následující omezení:

- Připojení musí být provedeno pomocí jedné z voleb MQCNO_RECONNECT_ * pro souvislou operaci. Tento proces znamená, že ačkoli může být komunikační relace přerušena, skutečný manipulátor připojení není přerušen a je místo toho ve stavu opětovného připojení. Pokud se opětovné připojení nezdaří, manipulátor připojení je nyní přerušen, což brání všem dalším voláním MQI.
- V tomto režimu jsou podporovány pouze MQPUT, MQGET, MQINQ a Async Consouběžné. Všechna příkazová slova MQOPEN, MQCLOSE nebo MQDISC vyžadují k dokončení nového připojení ke správci front.
- Stavové toky do zastavení správce front; jakýkoli stav ve správci front může být zastaralý nebo chybí. To znamená, že klienti mohou odesílat a přijímat zprávy a ve správci front není znám žádný stav. Další informace viz: Monitorování aplikací výběrového vysílání

Převod dat v rozhraní MQI pro výběrové vysílání systému zpráv

Pomocí těchto informací můžete porozumět tomu, jak funguje převod dat pro systém zpráv výběrového vysílání IBM MQ .

IBM MQ Výběrové vysílání je sdílený protokol bez připojení, a proto není možné, aby každý klient zadával specifické požadavky na převod dat. Každý klient přihlášený ke stejnému proudu výběrového vysílání obdrží stejná binární data; proto, pokud je vyžadován převod dat IBM MQ , je převod proveden lokálně na každém klientovi.

Data jsou převedena na klientovi pro přenos výběrového vysílání IBM MQ . Je-li zadána volba **MQGMO_CONVERT** , je převod dat proveden podle požadavku. Uživatelsky definované formáty vyžadují uživatelskou proceduru pro převod dat nainstalovanou na klientovi; informace o tom, které knihovny jsou nyní v balících klienta a serveru, naleznete v části “Zápis uživatelských procedur převodu dat” na stránce 950 .

Informace o administraci převodu dat naleznete v tématu Povolení převodu dat pro systém zpráv výběrového vysílání.

Další informace o převodu dat viz Převod dat.

Další informace o uživatelských procedurách pro převod dat a `ClientExitPath`naleznete v části ClientExitCesta ke konfiguračnímu souboru klienta.

Vytváření sestav výjimek výběrového vysílání

Pomocí těchto informací získáte informace o IBM MQ obslužných rutinách událostí výběrového vysílání a o výjimkách výběrového vysílání IBM MQ .

Výběrové vysílání IBM MQ pomáhá při určování problémů voláním obslužné rutiny událostí pro hlášení událostí výběrového vysílání, které jsou hlášeny pomocí standardního mechanizmu obslužné rutiny událostí IBM MQ .

Jednotlivá událost výběrového vysílání může mít za následek volání více než jedné události IBM MQ , protože může existovat více manipulátorů připojení `MQHCONN` používajících stejný vysílač nebo přijímač výběrového vysílání. Avšak každá výjimka výběrového vysílání způsobí, že pro každé připojení IBM MQ bude volána pouze jedna obslužná rutina událostí.

Konstanta IBM MQ `MQCBDO_EVENT_CALL` umožňuje aplikacím registrovat zpětné volání pro příjem pouze událostí IBM MQ a konstanta `MQCBDO_MC_EVENT_CALL` umožňuje aplikacím registrovat zpětné volání pro příjem pouze událostí výběrového vysílání. Jsou-li použity obě konstanty, jsou přijaty oba typy událostí.

Vyžádání událostí výběrového vysílání

IBM MQ Události výběrového vysílání 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 pro pole cbd.Options zadána volba MQCBDO_MC_EVENT_CALL , je obslužná rutina událostí namísto událostí na úrovni připojení odesílána pouze pro události výběrového vysílání IBM MQ . Chcete-li požadovat, aby byly do obslužné rutiny událostí odeslány oba typy událostí, musí aplikace zadat konstantu MQCBDO_EVENT_CALL do pole cbd.Options a 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);
```

Pokud není použita žádná z těchto konstant, do obslužné rutiny událostí se odesílají pouze události na úrovni připojení.

Další informace o hodnotách pro pole Options viz Volby (MQLONG).

Formát události výběrového vysílání

IBM MQ Výjimky výběrového vysílání zahrnují některé podpůrné informace, které jsou vráceny v parametru **Buffer** funkce zpětného volání. Ukazatel **Buffer** ukazuje na pole ukazatelů a pole MQCBC.DataLength uvádí velikost pole v bajtech. První prvek pole vždy ukazuje na krátký textový popis události. V závislosti na typu události mohou být zadány další parametry. V následující tabulce jsou uvedeny výjimky:

Tabulka 155. Popisy kódů událostí výběrového vysílání

Kód události	**Popis**	**Další data**
MQMCEV_PACKET_LOSS	Nezotavitelná ztráta paketu	Počet ztracených paketů
MQMCEV_HEARTBEAT_TIMEOUT	Dlouhá absence řídicího paketu prezenčního signálu	Není k dispozici
KONFLIKT verzí MQMCEV_VERSION_CONFLICT	Příjem paketů novější verze protokolu	Není k dispozici
MQMCEV_RELIABILITY	Různé režimy spolehlivosti vysílače a přijímače	Není k dispozici
MQMCEV_CLOSED_TRANS	Přenos tématu je uzavřen 1 zdrojem	Není k dispozici
MQMCEV_STREAM_ERROR	Zjištěna chyba v proudu	Není k dispozici
MQMCEV_NEW_SOURCE	Nový zdroj začne vysílat na téma	Zdrojová struktura
MQMCEV_RECEIVE_QUEUE_TRIMMED	Pakety odebrané z PacketQ kvůli vypršení času nebo prostoru	Počet oříznutých paketů
MQMCEV_PACKET_LOSS_NACK_EXPIRE	Nezotavitelná ztráta paketu v důsledku vypršení platnosti NACK	Počet ztracených paketů

Tabulka 155. Popisy kódů událostí výběrového vysílání (pokračování)

Kód události	Popis	Další data
MQMCEV_ACK_RETRIES_PŘEKROČEN	Pakety odebrané z historie po překročení hodnoty **max_ack_retries**	Počet odebraných paketů
MQMCEV_STREAM_SUSPEND_NACK	Na proudu přijatém tímto tématem byla pozastavena činnost NACK.	ID proudu pozastavení Doba v milisekundách, po kterou je proud pozastaven
MQMCEV_STREAM_RESUME_NACK	NACK byly obnoveny poté, co byly pozastaveny na proudu	ID proudu
MQMCEV_STREAM_EXPELUJE	Proud přijatý tímto tématem byl odmítnut kvůli požadavku na vyloučení	ID proudu
MQMCEV_FIRST_MESSAGE	První zpráva ze zdroje	Číslo zprávy.
MQMCEV_LATE_JOIN_FAILURE	Nezdařilo se spustit relaci pozdního spojení	Není k dispozici
MQMCEV_MESSAGE_LOSS	Nezotavitelná ztráta zprávy	Počet ztracených zpráv
MQMCEV_SEND_PACKET_FAILURE	Vysílači výběrového vysílání se nepodařilo odeslat paket výběrového vysílání.	Není k dispozici
MQMCEV_REPAIR_DELAY	Příjemce výběrového vysílání nepřijal opravný paket pro neprovedenou NAK	Není k dispozici
MQMCEV_MEMORY_ALERT_ON	Přijímací vyrovnávací paměti přijímače se zaplňují	Procento využití fondu vyrovnávacích pamětí
MQMCEV_MEMORY_ALERT_OFF	Přijímací vyrovnávací paměti přijímače jsou vypnuté na normální	Procento využití fondu vyrovnávacích pamětí
MQMCEV_NACK_ALERT_ON	Počet požadavků na opravu paketu přijímače dosáhl nejvyšší dosažitelné hodnoty	Aktuální četnost požadavků na opravu v paketech za sekundu
MQMCEV_NACK_ALERT_OFF	Četnost požadavků na opravu paketu příjemce je nižší než normální	Aktuální četnost požadavků na opravu v paketech za sekundu
MQMCEV_REPAIR_ALERT_ON	Rychlost odesílání paketu pro opravu vysílače dosáhla nejvyšší hodnoty	Není k dispozici
MQMCEV_REPAIR_ALERT_OFF	Rychlost odesílání paketu opravy vysílače je nižší než normální	Není k dispozici
MQMCEV_SHM_DEST_UNUSABLE	Bylo zjištěno, že oblast sdílené paměti používaná místem určení tématu vysílače je nepoužitelná.	Není k dispozici
MQMCEV_SHM_PORT_UNUSABLE	Bylo zjištěno, že port sdílené paměti používaný instancí příjemce je nepoužitelný.	Není k dispozici

Tabulka 155. Popisy kódů událostí výběrového vysílání (pokračování)		
Kód události	**Popis**	**Další data**
MQMCEV_CCT_GETTIME_FAILED	Čas získání z koordinovaného času klastru se nezdařil	Není k dispozici
MQMCEV_DEST_INTERFACE_FAILURE	Síťové rozhraní používané místem určení tématu vysílače se nezdařilo a záložní síťové rozhraní není k dispozici.	
MQMCEV_DEST_INTERFACE_FAILOVER	Síťové rozhraní používané místem určení tématu vysílače selhalo a bylo úspěšně překonáno selhání jiného rozhraní.	
MQMCEV_PORT_INTERFACE-SELHÁNÍ	Síťové rozhraní používané přijímačem rmmPort selhalo a záložní síťové rozhraní je nedostupné (nebo také selhalo).	RMM
MQMCEV_PORT_INTERFACE_FAILOVER	Síťové rozhraní používané přijímačem rmmPort selhalo a úspěšné překonání selhání na jiné rozhraní bylo dokončeno.	RMM

Kódování v C

Při kódování programů IBM MQ v jazyce C si povšimněte informací v následujících sekcích.

Parametry volání MQI

Parametry, které jsou *pouze pro vstup* a typu MQHCONN, MQHOBJ, MQHMSG nebo MQLONG, jsou předávány hodnotou; pro všechny ostatní parametry je *adresa* parametru předána hodnotou.

Ne všechny parametry předávané adresou je třeba zadat při každém vyvolání funkce. Pokud není konkrétní parametr povinný, může být jako parametr ve vyvolání funkce místo adresy dat parametru uveden nulový ukazatel. Parametry, pro které je to možné, jsou identifikovány v popisech volání.

Jako hodnota funkce není vrácen žádný parametr; v terminologii C to znamená, že všechny funkce vrací void.

Atributy funkce jsou definovány pomocí proměnné makra MQENTRY; hodnota této proměnné makra závisí na prostředí.

Parametry s nedefinovaným datovým typem

Funkce MQGET, MQPUT a MQPUT1 mají každý parametr **Buffer** , který má nedefinovaný datový typ. Tento parametr se používá k odeslání a přijetí dat zprávy aplikace.

Parametry tohoto řazení jsou zobrazeny v příkladech jazyka C jako pole MQBYTE. Tímto způsobem můžete deklarovat parametry, ale obvykle je výhodnější je deklarovat jako strukturu, která popisuje rozvržení dat ve zprávě. Parametr funkce je deklarován jako ukazatel-na-void, a proto může být adresa libovolných dat uvedena jako parametr při vyvolání funkce.

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 elementárního datového typu nebo datového typu struktury s předponou s písmenem P pro označení ukazatele. Atributy ukazatele jsou definovány pomocí proměnné makra MQPOINTER; hodnota této proměnné makra závisí na 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, porovnáváte nebo nastavujete 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 nefungují správně s daty deklarovanými jako MQBYTE24.

Manipulace se znakovými řetězci

Když správce front vrátí znaková data aplikaci, správce front vždy vyplní znaková data mezerami do definované délky pole. Správce front nevrací řetězce ukončené hodnotou null, ale můžete je použít ve vašem vstupu. Proto při kopírování, porovnávání nebo zřetězení takových řetězců použijte řetězcové funkce strncpy, strncmp nebo strncat.

Nepoužívejte řetězcové funkce, které vyžadují ukončení řetězce hodnotou null (strcpy, strcmp a strcat). Také nepoužívejte funkci strlen k určení délky řetězce; použijte místo toho funkci sizeof k určení délky pole.

Počáteční hodnoty pro struktury

Soubor začlenění <cmqc.h> definuje různé proměnné maker, které můžete použít k poskytnutí počátečních 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 definuje rozhraní MQI konkrétní hodnoty, které jsou platné (například pro pole *StrucId* nebo pro pole *Format* v deskriptoru MQMD). Pro každou z platných hodnot jsou poskytnuty dvě proměnné makra:

- Jedna proměnná makra definuje hodnotu jako řetězec s délkou, bez implicitní hodnoty null, která přesně odpovídá definované délce pole. V následujících příkladech představuje symbol ¬ jeden prázdný znak:

  ```
  #define MQMD_STRUC_ID "MD¬¬"
  #define MQFMT_STRING "MQSTR¬¬¬"
  ```

  Použijte tento formulář s funkcemi memcpy a memcmp.
- Druhá proměnná makra definuje hodnotu jako pole znaků; název této proměnné makra je název řetězcové formy s příponou _ARRAY. Příklad:

  ```
  #define MQMD_STRUC_ID_ARRAY 'M','D','¬','¬'
  #define MQFMT_STRING_ARRAY 'M','Q','S','T','R','¬','¬','¬'
  ```

  Pomocí tohoto formuláře inicializujte pole, když je instance struktury deklarována s hodnotami odlišnými od hodnot poskytnutých proměnnou makra MQMD_DEFAULT.

Počáteční hodnoty pro dynamické struktury

Je-li vyžadován proměnný počet instancí struktury, jsou instance obvykle vytvořeny v hlavní paměti získané dynamicky pomocí funkcí calloc nebo malloc.

Chcete-li inicializovat pole v těchto strukturách, doporučuje se následující technika:

1. Deklarujte instanci struktury pomocí příslušné proměnné makra MQxxx_DEFAULT pro inicializaci struktury. Tato instance se stane *modelem* pro ostatní instance:

   ```
   MQMD ModelMsgDesc = {MQMD_DEFAULT};
                                    /* declare model instance */
   ```

   Podle potřeby kódujte statická nebo automatická klíčová slova v deklaraci, abyste poskytli instanci modelu statickou nebo dynamickou dobu životnosti.
2. Chcete-li získat paměť pro dynamickou instanci struktury, použijte funkce calloc nebo malloc:

   ```
   PMQMD InstancePtr;
   InstancePtr = malloc(sizeof(MQMD));
                                    /* get storage for dynamic instance */
   ```
3. Pomocí funkce memcpy zkopírujte instanci modelu do dynamické instance:

   ```
   memcpy(InstancePtr,&ModelMsgDesc,sizeof(MQMD));
                                    /* initialize dynamic instance */
   ```

Použít z C++

Pro programovací jazyk C++ obsahují hlavičkové soubory následující další příkazy, které jsou zahrnuty pouze při použití kompilátoru C + +:

```
#ifdef __cplusplus
  extern "C" {
#endif

/* rest of header file */

#ifdef __cplusplus
  }
#endif
```

Windows Kódování v souboru Visual Basic

Informace, které je třeba vzít v úvahu při kódování programů IBM MQ v Microsoft Visual Basic. Parametr Visual Basic je podporován pouze v systému Windows.

Poznámka:

Stabilized V prostředí IBM WebSphere MQ 7.0mimo prostředí .NET byla podpora pro produkt Visual Basic (VB) stabilizována na úrovni IBM WebSphere MQ 6.0 . Většina nových funkcí přidaných do produktu IBM WebSphere MQ 7.0 nebo novějších není k dispozici pro aplikace VB. Pokud programujete ve VB.NET, použijte třídy IBM MQ pro .NET. Další informace viz téma Vývoj aplikací .NET.

Deprecated V produktu IBM MQ 9.0je podpora pro Microsoft Visual Basic 6.0 zamítnuta. Třídy IBM MQ pro .NET jsou doporučenými náhradními technologiemi.

Chcete-li se vyhnout nechtěnému převodu binárních dat předávaných mezi Visual Basic a IBM MQ, použijte namísto MQSTRING definici MQBYTE. CMQB.BAS definuje několik nových typů MQBYTE, které jsou ekvivalentní definici bajtu C a používají je v rámci struktur IBM MQ . Například pro strukturu MQMD (deskriptor zprávy) je MsgId (identifikátor zprávy) definován jako MQBYTE24.

Produkt Visual Basic nemá datový typ ukazatele, takže odkazy na jiné IBM MQ datové struktury jsou spíše posunutím než ukazatelem. Deklarujte složenou strukturu sestávající ze dvou struktur komponent a určete složenou strukturu na volání. IBM MQ support for Visual Basic (Podpora pro) poskytuje volání MQCONNXAny, které toto umožňuje a umožňuje klientským aplikacím určit vlastnosti kanálu v připojení klienta. Namísto typické struktury MQCNO přijímá netypovanou strukturu (MQCNOCD).

Struktura MQCNOCD je složená struktura sestávající z MQCNO následované MQCD. Tato struktura je deklarována v souboru záhlaví uživatelských procedur CMQXB. K inicializaci struktury MQCNOCD použijte rutinu MQCNOCD_DEFAULTS. K dispozici je ukázka volání MQCONNX (amqscnxb.vbp).

MQCONNXAny má stejné parametry jako MQCONNX s tím rozdílem, že parametr **ConnectOpts** je deklarován jako datový typ Any a nikoli jako datový typ MQCNO. To umožňuje funkci přijmout strukturu MQCNO nebo MQCNOCD. Tato funkce je deklarována v hlavičkovém souboru CMQB.

Související pojmy

“Příprava programů Visual Basic v adresáři Windows” na stránce 986
Informace, které je třeba zvážit při používání programů Microsoft Visual Basic na systému Windows.

Související odkazy

“Propojení aplikací Visual Basic s kódem IBM MQ MQI client” na stránce 888
Aplikace Microsoft Visual Basic můžete propojit s kódem IBM MQ MQI client na systému Windows.

Kódování v COBOLu

Při kódování programů IBM MQ v COBOLu si povšimněte informací v následující části.

Pojmenované konstanty

Zobrazí se názvy konstant obsahující znak podtržítka (_) jako součást názvu. V COBOLu musíte místo podtržítka použít znak pomlčky (-). Konstanty, které mají hodnoty znakových řetězců, používají znak apostrofu (') jako oddělovač řetězců. K tomu, aby kompilátor přijal tento znak, použijte volbu kompilátoru APOST.

Kopírovaný soubor CMQV obsahuje deklarace pojmenovaných konstant jako level-10 položek. Chcete-li použít konstanty, deklarujte položku level-01 explicitně, pak použijte příkaz COPY ke kopírování v deklaracích konstant:

```
WORKING-STORAGE SECTION.
01 MQM-CONSTANTS.
COPY CMQV.
```

Tato metoda však způsobí, že konstanty zabírají paměť v programu, i když se na ně neodkazuje. Pokud jsou konstanty zahrnuty v mnoha oddělených programech v rámci stejné jednotky spuštění, bude existovat více kopií konstant; to může vést k použití významného množství hlavní paměti. Tomu se můžete vyhnout přidáním klauzule GLOBAL do deklarace level-01 :

```
* Declare a global structure to hold the constants
01 MQM-CONSTANTS GLOBAL.
COPY CMQV.
```

To alokuje paměť pouze pro *jednu* sadu konstant v rámci jednotky spuštění; na tyto konstanty však může odkazovat *libovolný* program v rámci jednotky spuštění, nikoli pouze program, který obsahuje deklaraci level-01 .

Zajištění zarovnání struktury

Je třeba dbát na to, aby struktury IBM MQ , které se předávají ke spuštění volání produktu MQ , byly zarovnány na hranici slov. Slovní hranice je 4 bajty pro 32bitové procesy, 8 bajtů pro 64bitové procesy a 16 bajtů pro 128bitové procesy (IBM i).

Kde je to možné, umístěte všechny struktury IBM MQ dohromady tak, aby byly všechny zarovnané.

Kódování v jazyku System/390 assembleru (rozhraní fronty zpráv)

Při kódování programů IBM MQ for z/OS v jazyku asembleru si všimněte informací v následujících sekcích.

- “Názvy” na stránce 1019
- “Použití volání MQI” na stránce 1020
- “Deklarování konstant” na stránce 1020
- “Určení názvu struktury” na stránce 1020
- “Určení tvaru struktury” na stránce 1021
- “Řízení výpisu” na stránce 1021
- “Určení počátečních hodnot pro pole” na stránce 1021
- “Psaní reenterovatelných programů” na stránce 1021
- “Použití CEDF” na stránce 1022

Názvy

Názvy parametrů v popisech volání a názvy polí v popisech struktur jsou zobrazeny s velkými i velkými písmeny. V makrech jazyka asembleru dodaných s produktem IBM MQjsou všechny názvy uvedeny velkými písmeny.

Použití volání MQI

Rozhraní MQI je rozhraní volání, takže programy v jazyku assembleru musí dodržovat konvenci sestavení operačního systému.

Zejména před vydáním volání MQI musí programy v jazyku assembleru odkazovat na registraci R13 v oblasti ukládání alespoň 18 celých slov. Tato oblast uložení poskytuje paměť pro volaný program. Ukládá registry volajícího před zničením jejich obsahu a při návratu obnovuje obsah registrů volajícího.

Poznámka: To je důležité pro programy v jazyce CICS , které používají makro DFHEIENT k nastavení dynamického úložiště, ale které se rozhodnou přepsat výchozí DATAREG z R13 na jiné registry. Když rozhraní CICS Resource Manager přijme řízení ze stubu, uloží aktuální obsah registrů na adresu, na kterou ukazuje R13 . Neschopnost vyhradit oblast uložení pro tento účel poskytuje nepředvídatelné výsledky a pravděpodobně způsobí nevhodné ukončení v souboru CICS.

Deklarování konstant

Většina konstant je deklarována jako rovnítko v makru CMQA.

Následující konstanty však nelze definovat jako rovnítko a nejsou zahrnuty při volání makra s použitím výchozích voleb:

- MQACT_NONE
- MQCI_NONE
- MQFMT_NONE
- MQFMT_ADMIN
- MQFMT_COMMAND_1
- MQFMT_COMMAND_2
- MQFMT_DEAD_LETTER_HEADER
- MQFMT_EVENT
- MQFMT_IMS
- MQFMT_IMS_VAR_STRING
- MQFMT_PCF
- MQFMT_STRING
- MQFMT_TRIGGER
- MQFMT_XMIT_Q_HEADER
- MQMI_NONE

Chcete-li je zahrnout, přidejte při volání makra klíčové slovo EQUONLY=NO.

Produkt CMQA je chráněn proti vícenásobné deklaraci, takže jej můžete zahrnout mnohokrát. Klíčové slovo EQUONLY se však projeví pouze při prvním zahrnutí makra.

Určení názvu struktury

Chcete-li povolit deklaraci více než jedné instance struktury, makro, které generuje strukturu, předponou názvu každého pole je řetězec určený uživatelem a znak podtržítka (_).

Zadejte řetězec při vyvolání makra. Pokud nezadáte řetězec, makro použije název struktury k vytvoření předpony:

```
* Declare two object descriptors
CMQODA         Prefix used="MQOD_" (the default)
MY_MQOD CMQODA         Prefix used="MY_MQOD_"
```

Deklarace struktury v poli Popisy volání zobrazují výchozí předponu.

Určení tvaru struktury

Makra mohou generovat deklarace struktury v jedné ze dvou forem řízených parametrem DSECT:

DSECT = YES
Instrukce DSECT v jazyku asembleru se používá ke spuštění nové datové sekce; definice struktury bezprostředně následuje za příkazem DSECT. Není přidělena žádná paměť, takže není možná žádná inicializace. Popisek pro vyvolání makra se používá jako název datové sekce; není-li zadán žádný popisek, použije se název struktury.

DSECT = NO
Instrukce DC v jazyku asembleru se používají k definování struktury na aktuální pozici v rutině. Pole jsou inicializována pomocí hodnot, které lze určit kódováním příslušných parametrů při vyvolání makra. Pole, pro která nejsou při vyvolání makra zadány žádné hodnoty, jsou inicializována s výchozími hodnotami.

DSECT = NO se předpokládá, pokud není uveden parametr DSECT.

Řízení výpisu

Vzhled deklarace struktury ve výpisu assembleru můžete řídit pomocí parametru LIST:

LIST = YES
Deklarace struktury se zobrazí ve výpisu assembleru.

LIST = NO
Deklarace struktury se neobjevuje ve výpisu assembleru. Předpokládá se, že parametr LIST není zadán.

Určení počátečních hodnot pro pole

Můžete zadat hodnotu, která se má použít k inicializaci pole ve struktuře, tak, že jako parametr při vyvolání makra uvedete název tohoto pole (bez předpony) spolu s požadovanou hodnotou.

Chcete-li například deklarovat strukturu deskriptoru zpráv s polem *MsgType* inicializovaným pomocí MQMT_REQUEST a s polem *ReplyToQ* inicializovaným pomocí řetězce MY_REPLY_TO_QUEUE, použijte následující kód:

```
MY_MQMD    CMQMDA      MSGTYPE=MQMT_REQUEST,       X
REPLYTOQ=MY_REPLY_TO_QUEUE
```

Zadáte-li pojmenovanou konstantu (nebo rovnítko) jako hodnotu při vyvolání makra, definujte pojmenovanou konstantu pomocí makra CMQA. Hodnoty v apostrofech (''), které jsou znakové řetězce, nesmíte uzavřít do uvozovek.

Psaní reenterovatelných programů

Produkt IBM MQ používá své struktury pro vstup i výstup. Pokud chcete, aby váš program zůstal znovu zadatelný:

1. Definujte pracovní verze úložiště struktur jako DSECT nebo definujte struktury vložené v rámci již definovaného DSECT. Pak zkopírujte DSECT do úložiště, které je získáno pomocí:
   - Pro dávkové programy a programy TSO makra assembleru STORAGE nebo GETMAIN z/OS
   - V případě systému CICSse jedná o pracovní úložný prostor DSECT (DFHEISTG) nebo příkaz EXEC CICS GETMAIN.

   Chcete-li správně inicializovat tyto struktury pracovního úložiště, zkopírujte konstantní verzi odpovídající struktury do verze pracovního úložiště.

   Poznámka: Struktury MQMD a MQXQH jsou delší než 256 bajtů. Chcete-li tyto struktury zkopírovat do úložiště, použijte instrukci sestavovacího modulu MVCL.

2. Rezervujte prostor v úložišti pomocí formuláře LIST (MF=L) makra CALL. Při použití makra CALL k provedení volání MQI použijte formulář EXECUTE (MF=E) makra s použitím dříve vyhrazeného úložiště, jak ukazuje příklad v části “Použití CEDF” na stránce 1022. Další příklady, jak to provést, naleznete v ukázkových programech jazyka assembleru, které jsou dodávány s produktem IBM MQ.

Použijte volbu RENT jazyka asembleru, která vám pomůže určit, zda je váš program znovu zadatelný.

Informace o psaní reenterovatelných programů naleznete v příručce *z/OS MVS Application Development Guide: Assembler Language Programs*.

Použití CEDF

Chcete-li použít transakci dodávanou společností CICS, CEDF (CICS Execution Diagnostic Facility), která vám pomůže ladit váš program, přidejte ke každému příkazu CALL klíčové slovo ,VL , například:

```
CALL MQCONN,(NAME,HCONN,COMPCODE,REASON),MF=(E,PARMAREA),VL
```

Předchozí příklad je znovu zadatelný kód assembleru, kde PARMAREA je oblast v pracovním úložišti, kterou jste uvedli.

Použití volání MQI

Rozhraní MQI je rozhraní volání, takže programy v jazyku assembleru musí dodržovat konvenci sestavení operačního systému. Zejména před vydáním volání MQI musí programy v jazyku assembleru odkazovat na registraci R13 v oblasti ukládání alespoň 18 celých slov. Tato oblast uložení poskytuje paměť pro volaný program. Ukládá registry volajícího před zničením jejich obsahu a při návratu obnovuje obsah registrů volajícího.

Poznámka: To je důležité pro programy v jazyce CICS , které používají makro DFHEIENT k nastavení dynamického úložiště, ale které se rozhodnou přepsat výchozí DATAREG z R13 na jiné registry. Když rozhraní CICS Resource Manager přijme řízení ze stubu, uloží aktuální obsah registrů na adresu, na kterou ukazuje R13 . Neschopnost vyhradit vhodnou oblast uložení pro tento účel poskytuje nepředvídatelné výsledky a pravděpodobně způsobí nevhodné ukončení v souboru CICS.

IBM i Kódování IBM MQ programů v RPG (pouzeIBM i)

V dokumentaci k produktu IBM MQ jsou parametry volání, názvy datových typů, pole struktur a názvy konstant popsány pomocí jejich dlouhých názvů. V RPG jsou tyto názvy zkráceny na šest nebo méně velkých písmen.

Například pole *MsgType* se v RPG stane *MDMT* . Další informace naleznete v příručce IBM i Application Programming Reference (ILE/RPG).

Kódování v PL/I (pouzez/OS)

Užitečné informace při kódování pro IBM MQ v PL/I.

Struktury

Struktury jsou deklarovány s atributem BASED, a proto nezabírají žádnou paměť, pokud program nedeklaruje jednu nebo více instancí struktury.

Instanci struktury lze deklarovat 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; když se atribut like používá k deklaraci instance struktury, tato instance zdědí počáteční hodnoty definované pro tuto strukturu. Musíte nastavit pouze ta pole, kde se požadovaná hodnota liší od počáteční hodnoty.

Jazyk PL/I nerozlišuje malá a velká písmena, takže názvy volání, strukturovaná pole a konstanty lze kódovat malými písmeny, velkými písmeny nebo smíšenými písmeny.

Pojmenované konstanty

Pojmenované konstanty jsou deklarovány jako proměnné makra; v důsledku toho pojmenované konstanty, na které program neodkazuje, nezabírají v kompilované proceduře žádnou paměť.

Avšak volba kompilátoru, která způsobí, že zdroj bude zpracován preprocesorem maker, musí být uvedena při kompilaci programu.

Všechny proměnné makra jsou znakové proměnné, dokonce i ty, které představují číselné hodnoty. Přestože se může zdát, že se jedná o čítač intuitivní, nevede k žádnému konfliktu datových typů po nahrazení proměnných makra procesorem maker, například:

```
%dcl MQMD_STRUC_ID char;
%MQMD_STRUC_ID = '''MD ''';

%dcl MQMD_VERSION_1 char;
%MQMD_VERSION_1 = '1';
```

Použití ukázkových procedurálních programů IBM MQ

Tyto ukázkové programy jsou napsány v procedurálních jazycích a ukazují typické použití rozhraní MQI (Message Queue Interface). Programy IBM MQ na různých platformách.

Informace o této úloze

Existují dvě sady vzorků:

- Ukázkové programy pro distribuované systémy a IBM i.
- Ukázkové programy pro z/OS.

Procedura

- Chcete-li se dozvědět více o ukázkových programech, použijte následující odkazy:
  - “Použití ukázkových programů na platformě Multiplatforms” na stránce 1024
  - z/OS “Použití ukázkových programů pro z/OS” na stránce 1122

Související pojmy

“Koncepty vývoje aplikací” na stránce 7
K psaní aplikací IBM MQ můžete použít výběr procedurálních nebo objektově orientovaných jazyků. Než začnete navrhovat a psát aplikace IBM MQ , seznamte se se základními koncepty produktu IBM MQ .

“Vyvíjení aplikací pro IBM MQ” na stránce 5
Můžete vyvíjet aplikace pro odesílání a příjem zpráv a pro správu správců front a souvisejících prostředků. Produkt IBM MQ podporuje aplikace napsané v mnoha různých jazycích a rámcích.

“Aspekty návrhu pro aplikace IBM MQ” na stránce 47
Když jste se rozhodli, jak mohou vaše aplikace využívat platformy a prostředí, která máte k dispozici, musíte se rozhodnout, jak používat funkce nabízené produktem IBM MQ.

“Psaní procedurální žádosti o zařazení do fronty” na stránce 699
Prostřednictvím těchto informací můžete získat informace o psaní aplikací řazení do front, o připojování a odpojování od správce front, o publikování/odběru a o otevírání a zavírání objektů.

“Psaní procedurálních aplikací klienta” na stránce 882
Co potřebujete vědět, abyste mohli psát klientské aplikace na systému IBM MQ pomocí procedurálního jazyka.

“Zápis aplikací publikování/odběru” na stránce 782

Začněte psát aplikace publikování/odběru IBM MQ .

“Sestavení procedurální aplikace” na stránce 967
Aplikaci IBM MQ můžete napsat v jednom z několika procedurálních jazyků a spustit ji na několika různých platformách.

“Zpracování chyb procedurálních programů” na stránce 1004
Tyto informace vysvětlují chyby související s voláními rozhraní MQI aplikací, a to buď při volání, nebo při doručení zprávy do konečného místa určení.

Multi Použití ukázkových programů na platformě Multiplatforms

Tyto ukázkové procedurální programy jsou dodávány s produktem. Ukázky jsou napsány v jazycích C a COBOL a ukazují typické použití rozhraní MQI (Message Queue Interface).

Informace o této úloze

Ukázky nejsou určeny k demonstraci obecných programovacích technik, takže některé kontroly chyb, které byste mohli chtít zahrnout do produkčního programu, jsou vynechány.

Zdrojový kód všech ukázek je poskytován s produktem; tento zdroj obsahuje komentáře, které vysvětlují techniky řazení zpráv do front, které jsou demonstrovány v programech.

IBM i Informace o programování v jazyce RPG naleznete v tématu IBM i Referenční příručka pro programování aplikací (ILE/RPG).

Názvy ukázek začínají předponou `amq`. Čtvrtý znak označuje programovací jazyk a kompilátor v případě potřeby:

- s: Jazyk C
- 0: Jazyk COBOL na kompilátorech IBM i Micro Focus
- i: Jazyk COBOL pouze na kompilátorech IBM
- m: Jazyk COBOL pouze pro kompilátory Micro Focus

Osmý znak spustitelného souboru označuje, zda je ukázka spuštěna v režimu lokální vazby nebo v režimu klienta. Pokud neexistuje žádný osmý znak, spustí se ukázka v režimu lokálních vazeb. Je-li osmý znak ' c', spustí se ukázka v režimu klienta.

Než budete moci spustit ukázkové aplikace, musíte nejprve vytvořit a nakonfigurovat správce front. Chcete-li nastavit správce front tak, aby přijímal připojení klienta, viz “Konfigurace správce front pro přijetí připojení klienta na platformě Multiplatforms” na stránce 1034.

Procedura

- Chcete-li se dozvědět více o ukázkových programech, použijte následující odkazy:
  - “Funkce demonstrované v ukázkových programech na platformě Multiplatforms” na stránce 1025
  - “Příprava a spuštění ukázkových programů” na stránce 1033
  - “Ukázkový program uživatelské procedury rozhraní API” na stránce 1039
  - “Ukázkový program Asynchronní spotřeba” na stránce 1040
  - “Ukázkový program Asynchronní vložení” na stránce 1041
  - “Ukázkové programy Procházet” na stránce 1042
  - “Ukázkový program prohlížeče” na stránce 1043
  - “Ukázka transakce CICS” na stránce 1044
  - “Ukázkový program Connect” na stránce 1045
  - “Ukázkový program Data-Conversion” na stránce 1046
  - “Ukázky koordinace databáze” na stránce 1046

- “Ukázka obslužné rutiny fronty nedoručených zpráv” na stránce 1053
- “Ukázkový program distribučního seznamu” na stránce 1053
- “Ukázkové programy Echo” na stránce 1054
- “Ukázkové programy Get” na stránce 1055
- “Ukázkové programy s vysokou dostupností” na stránce 1056
- “Ukázkové programy dotazování” na stránce 1060
- “Vlastnosti dotazu ukázkového programu obslužné rutiny zpráv” na stránce 1061
- “Ukázkové programy publikování/odběru” na stránce 1061
- “Ukázkový program uživatelské procedury publikování” na stránce 1066
- “Ukázkové programy Put” na stránce 1067
- “Ukázkové programy referenčních zpráv” na stránce 1069
- “Ukázkové programy požadavků” na stránce 1076
- “Ukázkové programy Set” na stránce 1081
- “Ukázkový program TLS” na stránce 1082
- “Ukázkové programy spuštění” na stránce 1085
- “Použití ukázek TUXEDO v systému AIX, Linux, and Windows” na stránce 1087
- “Použití uživatelské procedury zabezpečení rozhraní SSPI v systému Windows” na stránce 1096
- “Spuštění ukázek pomocí vzdálených front” na stránce 1097
- “Ukázkový program pro monitorování front klastru (AMQSCLM)” na stránce 1097
- “Ukázkový program pro vyhledání koncového bodu připojení (CEPL)” na stránce 1107

Související pojmy

“Ukázkové programy C++” na stránce 515
K dispozici jsou čtyři ukázkové programy, které demonstrují získávání a vkládání zpráv.

Související úlohy

“Použití ukázkových programů pro z/OS” na stránce 1122
Ukázkové procedurální aplikace dodávané s produktem IBM MQ for z/OS demonstrují typické použití rozhraní MQI (Message Queue Interface).

Multi *Funkce demonstrované v ukázkových programech na platformě Multiplatforms*

Kolekce tabulek, které ukazují techniky demonstrované ukázkovými programy IBM MQ .

Všechny ukázky otevírají a zavírají fronty pomocí volání MQOPEN a MQCLOSE, takže tyto techniky nejsou v tabulkách uvedeny odděleně. Viz záhlaví, které obsahuje platformu, o kterou máte zájem.

z/OS Pro platformu z/OS viz “Použití ukázkových programů pro z/OS” na stránce 1122.

Linux AIX *Ukázky pro systémy AIX and Linux*
Techniky demonstrované ukázkovými programy pro IBM MQ for AIX or Linux.

Chcete-li zjistit, kde jsou uloženy ukázkové programy pro produkt IBM MQ for AIX or Linux , prohlédněte si téma “Příprava a spuštění ukázkových programů na systému AIX and Linux” na stránce 1037 .

Tabulka 156 na stránce 1026 Tabulka uvádí, které zdrojové soubory C a COBOL jsou poskytnuty, a zda je zahrnut spustitelný soubor serveru nebo klienta.

Tabulka 156. Ukázkové programy, které demonstrují použití rozhraní MQI (C a COBOL) na systému AIX and Linux.

Tabulka se čtyřmi sloupci. V prvních sloupcích jsou uvedeny techniky demonstrované ukázkami. Ve druhém sloupci jsou uvedeny ukázky jazyka C a ve třetím sloupci jsou uvedeny ukázky jazyka COBOL, které demonstrují jednotlivé techniky uvedené v prvním sloupci. Čtvrtý sloupec zobrazuje, zda je nebo není zahrnut spustitelný soubor C serveru, a pátý sloupec uvádí, zda spustitelný soubor C klienta je nebo není zahrnut.

Technika	**C (zdroj) ("1" na stránce 1028)**	**COBOL (zdroj) ("2" na stránce 1028)**	**Server (spustitelný soubor C)**	**Klient (spustitelný soubor C)**
Použití rozhraní publikování/odběru	amqspuba amqssuba amqssbxa	bez vzorku	amqspub amqssub amqssbx	bez vzorku
Vkládání 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 distribučního seznamu ("3" na stránce 1028)	amqsptl0	amq0ptl0.cbl	amqsptl	amqsptlc
Odpověď na zprávu požadavku	amqsinqa se	amqminqx amqiinqx	amqsinq	bez vzorku
Získávání zpráv pomocí procházení (bez čekání)	amqsgbr0	amq0gbr0	amqsgbr	bez vzorku
Získávání zpráv (čekat s časovým limitem)	amqsget0	amq0get0	amqsget	amqsgetc
Získávání zpráv (neomezené čekání)	amqstrg0	bez vzorku	amqstrg	amqstrgc
Získávání zpráv (s převodem dat)	amqsecha	bez vzorku	amqsech	bez vzorku
Vložení referenčních zpráv do fronty ("3" na stránce 1028)	amqsprma	bez vzorku	amqsprm	amqsprmc
Získání referenčních zpráv z fronty ("3" na stránce 1028)	amqsgrma	bez vzorku	amqsgrm	amqsgrmc
Uživatelská procedura kanálu referenčních zpráv ("3" na stránce 1028)	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 se	amqminqx amqiinqx	amqsinq	amqsinqc
Použití výlučné vstupní fronty	amqstrg0	amq0req0	amqstrg	amqstrgc
Použití volání MQINQ	amqsinqa se	amqminqx amqiinqx	amqsinq	bez vzorku
Použití volání MQSET	amqsseta	amqmsetx amqisetx	sada amqsset	amqssetc
Použití fronty pro odpověď	amqsreq0	amq0req0	amqsreq (požadavek na změnu)	amqsreqc-počet

Tabulka 156. Ukázkové programy, které demonstrují použití rozhraní MQI (C a COBOL) na systému AIX and Linux.

Tabulka se čtyřmi sloupci. V prvních sloupcích jsou uvedeny techniky demonstrované ukázkami. Ve druhém sloupci jsou uvedeny ukázky jazyka C a ve třetím sloupci jsou uvedeny ukázky jazyka COBOL, které demonstrují jednotlivé techniky uvedené v prvním sloupci. Čtvrtý sloupec zobrazuje, zda je nebo není zahrnut spustitelný soubor C serveru, a pátý sloupec uvádí, zda spustitelný soubor C klienta je nebo není zahrnut.

(pokračování)

Technika	**C (zdroj) (“1” na stránce 1028)**	**COBOL (zdroj) (“2” na stránce 1028)**	**Server (spustitelný soubor C)**	**Klient (spustitelný soubor C)**
Požadování výjimek zpráv	amqsreq0	amq0req0	amqsreq (požadavek na změnu)	bez vzorku
Přijetí zkrácené zprávy	amqsgbr0	amq0gbr0	amqsgbr	bez vzorku
Použití rozlišeného názvu fronty	amqsgbr0	amq0gbr0	amqsgbr	bez vzorku
Spuštění procesu	amqstrg0	bez vzorku	amqstrg	amqstrgc
Použití převodu dat	(“4” na stránce 1028)	bez vzorku	bez vzorku	bez vzorku
IBM MQ (koordinace správců databází vyhovujících standardu XA), kteří přistupují k jedné databázi pomocí jazyka SQL.	amqsxas0.sqc Db2 amqsxas0.ec Informix	amq0xas0.sqb	bez vzorku	bez vzorku
IBM MQ (koordinace správců databází vyhovujících standardu XA), kteří přistupují ke dvěma databázím pomocí jazyka SQL.	amqsxag0.c amqsxab0.sqc amqsxaf0.sqc	amq0xag0.cbl amq0xab0.sqb amq0xaf0.sqb	bez vzorku	bez vzorku
CICS transakce (“5” na stránce 1028)	amqscic0.ccs	bez vzorku	amqscic0	bez vzorku
Transakce Enčina (“3” na stránce 1028)	amqsxae0	bez vzorku	amqsxae0	bez vzorku
Transakce TUXEDO pro vložení zpráv “6” na stránce 1028)	amqstxpx	bez vzorku	bez vzorku	bez vzorku
Transakce TUXEDO pro získání zpráv (“6” na stránce 1028)	amqstxgx	bez vzorku	bez vzorku	bez vzorku
Server pro TUXEDO (“6” na stránce 1028)	amqstxsx	bez vzorku	bez vzorku	bez vzorku
obslužná rutina fronty nedoručených zpráv	Adresář `./tools/c/Samples/dlq` (“7” na stránce 1028)	bez vzorku	amqsdlq-počet	bez vzorku
Vložení zprávy z klienta MQI	bez vzorku	bez vzorku	bez vzorku	amqsputc
Získání zprávy z klienta MQI	bez vzorku	bez vzorku	bez vzorku	amqsgetc
Připojení ke správci front pomocí MQCONNX	amqscnxc	bez vzorku	bez vzorku	amqscnxc
Použití uživatelských procedur rozhraní API	amqsaxe0	bez vzorku	amqsaxe	bez vzorku

Tabulka 156. Ukázkové programy, které demonstrují použití rozhraní MQI (C a COBOL) na systému AIX and Linux.

Tabulka se čtyřmi sloupci. V prvních sloupcích jsou uvedeny techniky demonstrované ukázkami. Ve druhém sloupci jsou uvedeny ukázky jazyka C a ve třetím sloupci jsou uvedeny ukázky jazyka COBOL, které demonstrují jednotlivé techniky uvedené v prvním sloupci. Čtvrtý sloupec zobrazuje, zda je nebo není zahrnut spustitelný soubor C serveru, a pátý sloupec uvádí, zda spustitelný soubor C klienta je nebo není zahrnut.

(pokračování)

Technika	**C (zdroj) ("1" na stránce 1028)**	**COBOL (zdroj) ("2" na stránce 1028)**	**Server (spustitelný soubor C)**	**Klient (spustitelný soubor C)**
Ukončení vyrovnávání pracovní zátěže klastru	amqswlm0	bez vzorku	amqswlm	bez vzorku
Asynchronní vkládání zpráv a získávání stavu pomocí volání MQSTAT	amqsapt0	bez vzorku	amqsapt	amqsaptc
Opakovaně připojitelní klienti	amqsphac amqsghac amqsmhac	bez vzorku	nelze použít	amqsphac amqsghac amqsmhac
Použití spotřebitelů zpráv k asynchronnímu přijímání zpráv z více front	amqscbf0	bez vzorku	amqscbf	amqscbfc
Určení informací o připojení TLS v MQCONNX	amqssslc	bez vzorku	nelze použít	amqssslc

Notes:

1. Spustitelná verze ukázek IBM MQ MQI client sdílí stejný zdroj jako ukázky spuštěné v prostředí serveru.
2. Kompilační programy začínající 'amqm' kompilátorem Micro Focus COBOL, programy začínající 'amqi' kompilátorem IBM COBOL a programy začínající 'amq0' oboje.
3. AIX Podporováno pouze na systémech IBM MQ for AIX .
4. AIX V systémech IBM MQ for AIX se tento program nazývá `amqsvfc0.c`
5. AIX Produkt CICS je podporován pouze produkty IBM MQ for AIX .
6. Linux Produkt TUXEDO není podporován produktem IBM MQ pro systém Linux v systému System p.
7. Zdroj pro obslužnou rutinu fronty nedoručených zpráv se skládá z několika souborů a je uveden v odděleném adresáři.

Podrobné informace o podpoře pro systémy AIX and Linux viz Systémové požadavky pro IBM MQ.

Windows *Ukázky pro IBM MQ for Windows*

Techniky demonstrované ukázkovými programy pro IBM MQ for Windows.

Tabulka 157 na stránce 1029 uvádí, které zdrojové soubory C a COBOL jsou poskytnuty, a zda je zahrnut spustitelný soubor serveru nebo klienta.

Tabulka 157. Ukázkové programy IBM MQ for Windows předvádějící použití rozhraní MQI (C a COBOL)

Technika	C (zdroj)	COBOL (zdroj)	Server (spustitelný soubor C)	Klient (spustitelný soubor C)
Použití rozhraní publikování/odběru	amqspuba amqssuba amqssbxa	bez vzorku	amqspub amqssub amqssbx	bez vzorku
Vkládání zpráv pomocí volání MQPUT	amqsput0	amq0put0	amqsput	amqsputc
Vložení jedné zprávy pomocí volání MQPUT1	amqsinqa amqsecha	amqminq2 amqmech2 amqiinq2 amqiech2	amqsinq amqsech	amqsinqc amqsechc (amqsinqc)
Vložení zpráv do distribučního seznamu	amqsptl0	amq0ptl0.cbl	amqsptl	amqsptlc
Odpověď na zprávu požadavku	amqsinqa se	amqminq2 amqiinq2	amqsinq	amqsinqc
Získávání zpráv (bez čekání)	amqsgbr0	amq0gbr0	amqsgbr	amqsgbrc
Získávání zpráv (čekat s časovým limitem)	amqsget0	amq0get0	amqsget	amqsgetc
Získávání zpráv (neomezené čekání)	amqstrg0	bez vzorku	amqstrg	amqstrgc
Získávání zpráv (s převodem dat)	amqsecha	bez vzorku	amqsech	amqsechc
Vložení referenčních zpráv do fronty	amqsprma	bez vzorku	amqsprm	amqsprmc
Získání referenčních zpráv z fronty	amqsgrma	bez vzorku	amqsgrm	amqsgrmc
Uživatelská procedura kanálu referenčních 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 se	amqminq2 amqiinq2	amqsinq	amqsinqc
Použití výlučné vstupní fronty	amqstrg0	amq0req0	amqstrg	amqstrgc
Použití volání MQINQ	amqsinqa se	amqminq2 amqiinq2	amqsinq	amqsinqc
Použití volání MQSET	amqsseta	amqmset2 amqiset2	sada amqsset	amqssetc
Použití volání MQINQMP	amqsiqma	bez vzorku	bez vzorku	bez vzorku
Použití fronty pro odpověď	amqsreq0	amq0req0	amqsreq (požadavek na změnu)	amqsreqc-počet
Požadování výjimek zpráv	amqsreq0	amq0req0	amqsreq (požadavek na změnu)	amqsreqc-počet
Přijetí zkrácené zprávy	amqsgbr0	amq0gbr0	amqsgbr	amqsgbrc
Použití rozlišeného názvu fronty	amqsgbr0	amq0gbr0	amqsgbr	amqsgbrc
Spuštění procesu	amqstrg0	bez vzorku	amqstrg	amqstrgc

Tabulka 157. Ukázkové programy IBM MQ for Windows předvádějící použití rozhraní MQI (C a COBOL) (pokračování)

Technika	C (zdroj)	COBOL (zdroj)	Server (spustitelný soubor C)	Klient (spustitelný soubor C)
Použití převodu dat	amqsvfc0	bez vzorku	bez vzorku	bez vzorku
IBM MQ (koordinace správců databází vyhovujících standardu XA), kteří přistupují k jedné databázi pomocí jazyka SQL.	amqsxas0.sqc Db2 amqsxas0.ec Informix	amq0xas0.sqb	bez vzorku	bez vzorku
IBM MQ (koordinace správců databází vyhovujících standardu XA), kteří přistupují ke 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 1030)	bez vzorku	amqsdlq-počet	bez vzorku
Z IBM MQ MQI client, vložení zprávy	bez vzorku	bez vzorku	bez vzorku	amqsputc
Z IBM MQ MQI client, získání zprávy	bez vzorku	bez vzorku	bez vzorku	amqsgetc
Připojení ke správci front pomocí MQCONNX	amqscnxc	bez vzorku	bez vzorku	amqscnxc
Použití uživatelských procedur rozhraní API	amqsaxe0	bez vzorku	amqsaxe	bez vzorku
Vyrovnávání pracovní zátěže klastru	amqswlm0	bez vzorku	amqswlm	bez vzorku
Rutiny zabezpečení SSPI	amqsspinu	bez vzorku	amqrspin.dll	amqrspin.dll
Asynchronní vkládání zpráv a získávání stavu pomocí volání MQSTAT	amqsapt0	bez vzorku	amqsapt	amqsaptc
Opakovaně připojitelní klienti	amqsphac amqsghac amqsmhac	bez vzorku	Nelze použít	amqsphac amqsghac amqsmhac
Použití spotřebitelů zpráv k asynchronnímu přijímání zpráv z více front	amqscbf0	bez vzorku	amqscbf	amqscbfc
Určení informací o připojení TLS v MQCONNX	amqsssl c	bez vzorku	nelze použít	amqsssl c

Notes:

1. Zdroj pro obslužnou rutinu fronty nedoručených zpráv se skládá z několika souborů a je uveden v odděleném adresáři.

Windows *Ukázky produktu Visual Basic pro produkt IBM MQ for Windows*

Techniky demonstrované ukázkovými programy pro systémy IBM MQ na systému Windows .

Tabulka 158 na stránce 1031 ukazuje techniky demonstrované ukázkovými programy IBM MQ for Windows .

Projekt může obsahovat několik souborů. Když otevřete projekt v jazyku Visual Basic, ostatní soubory se načtou automaticky. Nejsou poskytnuty žádné spustitelné programy.

Všechny ukázkové projekty s výjimkou mqtrivc.vbpjsou nastaveny pro práci se serverem IBM MQ . Chcete-li zjistit, jak změnit ukázkové projekty tak, aby spolupracovaly s IBM MQ klienty, viz “Příprava programů Visual Basic v adresáři Windows” na stránce 986.

Tabulka 158. Ukázkové programy IBM MQ for Windows předvádějící použití rozhraní MQI (Visual Basic)

Technika	**Název souboru projektu**
Vkládání zpráv pomocí volání MQPUT	amqsputb.vbp
Získávání zpráv pomocí volání MQGET	amqsgetb.vbp
Procházení fronty pomocí volání MQGET	amqsbcgb.vbp
Jednoduchá ukázka MQGET a MQPUT (klient)	mqtrivc.vbp
Jednoduchá ukázka MQGET a MQPUT (server)	mqtrivs.vbp
Vkládání a získávání řetězců a struktur definovaných uživatelem pomocí MQPUT a MQGET	strings.vbp
Použití struktur PCF ke spuštění a zastavení kanálu	pcfsamp.vbp
Vytvoření fronty pomocí MQAI	amqsaicq.vbp
Zobrazení seznamu front správce front pomocí rozhraní MQAI	amqsailq.vbp
Monitorování událostí pomocí MQAI	amqsaiem.vbp

IBM i *Ukázky pro IBM i*

Techniky demonstrované ukázkovými programy pro systémy IBM MQ na systému IBM i .

Tabulka 159 na stránce 1031 ukazuje techniky demonstrované ukázkovými programy IBM MQ for IBM i . Některé techniky se vyskytují ve více než jednom ukázkovém programu, ale v tabulce je uveden pouze jeden program.

Tabulka 159. Ukázkové programy předvádějící použití rozhraní MQI (C a COBOL) na systému IBM i

Technika	**C (zdroj) (“1” na stránce 1033)**	**COBOL (zdroj) (“2” na stránce 1033)**	**RPG (zdroj) (“3” na stránce 1033)**	**Klient (spustitelný soubor C) (4)**
Vkládání zpráv pomocí volání MQPUT	AMQSPUT0	AMQ0PUT4	AMQ3PUT4	amqsputc
Vkládání 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	AMQSPTLC
Odpověď na zprávu požadavku	AMQSINQ4	AMQ0INQ4	AMQ3INQ4	AMQSINQC
Získávání zpráv (bez čekání)	AMQSGBR4	AMQ0GBR4	AMQ3GBR4	AMQSGBRC
Získávání zpráv (čekat s časovým limitem)	AMQSGET4	AMQ0GET4	AMQ3GET4	AMQSGETC

Tabulka 159. Ukázkové programy předvádějící použití rozhraní MQI (C a COBOL) na systému IBM i (pokračování)

Technika	**C (zdroj) ("1" na stránce 1033)**	**COBOL (zdroj) ("2" na stránce 1033)**	**RPG (zdroj) ("3" na stránce 1033)**	**Klient (spustitelný soubor C) (4)**
Získávání zpráv (neomezené čekání)	AMQSTRG4	bez vzorku	AMQ3TRG4	AMQSTRGC
Získávání zpráv (s převodem dat)	AMQSECH4	AMQ0ECH4	AMQ3ECH4	AMQSECHC
Vložení referenčních zpráv do fronty	AMQSPRM4	bez vzorku	bez vzorku	AMQSPRMC
Získání referenčních zpráv z fronty	AMQSGRM4	bez vzorku	bez vzorku	AMQSGRMC
Uživatelská procedura kanálu referenčních zpráv	AMQSQRM4, AMQSXRM4	bez vzorku	bez vzorku	bez vzorku
Ukončení zprávy	AMQSCMX4	bez vzorku	bez vzorku	bez vzorku
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	AMQSREQC
Použití volání MQINQ	AMQSINQ4	AMQ0INQ4	AMQ3INQ4	AMQSINQC
Použití volání MQSET	AMQSSET4	AMQ0SET4	AMQ3SET4	AMQSSETC
Použití fronty pro odpověď	AMQSREQ4	AMQ0REQ4	AMQ3REQ4	AMQSREQC
Požadování výjimek zpráv	AMQSREQ4	AMQ0REQ4	AMQ3REQ4	AMQSREQC
Přijetí zkrácené zprávy	AMQSGBR4	AMQ0GBR4	AMQ3GBR4	AMQSGBRC
Použití rozlišeného názvu fronty	AMQSGBR4	AMQ0GBR4	AMQ3GBR4	AMQSGBRC
Spuštění 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 převodu dat	AMQSVFC4	bez vzorku	bez vzorku	bez vzorku
Použití uživatelských procedur rozhraní API	AMQSAXE0	bez vzorku	bez vzorku	bez vzorku
Vyrovnávání pracovní zátěže klastru	AMQSWLM0	bez vzorku	bez vzorku	bez vzorku
Asynchronní vkládání zpráv a získávání stavu pomocí volání MQSTAT	AMQSAPT0	bez vzorku	bez vzorku	AMQSAPTC
Použití rozhraní publikování/odběru	AMQSPUBA, AMQSSUBA, AMQSSBXA	bez vzorku	bez vzorku	AMQSPUBC, AMQSSUBC, AMQSSBXC
Znovu připojitelní klienti (5)	AMQSPHAC, AMQSGHAC, AMQSMHAC	bez vzorku	bez vzorku	bez vzorku
Použití spotřebitelů zpráv k asynchronnímu přijímání zpráv z více front (5)	AMQSCBFO	bez vzorku	bez vzorku	bez vzorku
Určení informací o připojení TLS v MQCONNX	AMQSSSLC	bez vzorku	bez vzorku	AMQSSSLC

Tabulka 159. Ukázkové programy předvádějící použití rozhraní MQI (C a COBOL) na systému IBM i (pokračování)				
Technika	**C (zdroj) ("1" na stránce 1033)**	**COBOL (zdroj) ("2" na stránce 1033)**	**RPG (zdroj) ("3" na stránce 1033)**	**Klient (spustitelný soubor C) (4)**
Připojení ke správci front pomocí MQCONNX	AMQSCNXC	bez vzorku	bez vzorku	AMQSCNXC
Zjistit vlastnosti popisovače zprávy pomocí MQINQMP z fronty zpráv	AMQISQMA	bez vzorku	bez vzorku	AMQISQMC
Nastavení vlastností manipulátoru zprávy pomocí MQSETMP a jeho vložení do fronty zpráv	AMQSSQMA	bez vzorku	bez vzorku	AMQSSQMC

Notes:

1. Zdroj ukázek C je v souboru QMQMSAMP/QCSRC. Zahrnout soubory existující jako členy do souboru QMQM/H.
2. Zdroj pro ukázky v jazyce COBOL je v souborech QMQMSAMP/QCBLLESRC. Členové mají název AMQ0 xxx 4, kde xxx označuje ukázkovou funkci.
3. Zdroj pro ukázky RPG je v QMQMSAMP/QRPGLESRC. Členové mají název AMQ3 xxx 4, kde xxx označuje ukázkovou funkci. V QMQM/QRPGLESRC existují členové kopie. Každý název člena má příponu G.
4. Spustitelná verze ukázek IBM MQ MQI client sdílí stejný zdroj jako ukázky spuštěné v prostředí serveru. Zdroj ukázek v prostředí klienta je stejný jako server. Ukázky produktu IBM MQ MQI client jsou propojeny s knihovnou klienta LIBMQIC a ukázky serveru IBM MQ jsou propojeny s knihovnou serveru LIBMQM.
5. Pokud má být spuštěn spustitelný soubor klienta pro ukázkovou aplikaci klienta Reconnectable a asynchronně odběratelskou aplikaci, musí být kompilován a propojen s knihovnou podprocesů LIBMQIC_R. Proto musí být spuštěn v prostředí s podporou podprocesů. Nastavte proměnnou prostředí QIBM_MULTI_THREAD_NA 'Y' a spusťte aplikaci z qsh.

   Další informace viz Nastavení IBM MQ pomocí Java a JMS .

Další informace viz "Příprava a spuštění ukázkových programů na systému IBM i" na stránce 1036.

Kromě těchto možností obsahuje ukázková volba IBM MQ for IBM i ukázkový datový soubor, který použijete jako vstup do ukázkových programů AMQSDATA a ukázkových CL programů, které demonstrují úlohy administrace. Ukázky CL jsou popsány v tématu Administrace IBM i . Můžete použít ukázkový CL program amqsamp4 k vytvoření front pro použití s ukázkovými programy popsanými v tomto tématu.

Multi

Příprava a spuštění ukázkových programů

Po dokončení počáteční přípravy můžete spustit ukázkové programy.

Informace o této úloze

Před spuštěním ukázkových programů musíte nejprve vytvořit správce front a také fronty, které potřebujete. Možná budete také muset provést nějakou další přípravu, například pokud chcete spustit ukázky v jazyce COBOL. Po dokončení nezbytné přípravy můžete spustit ukázkové programy.

Postup

Chcete-li získat informace o tom, jak připravit a spustit ukázkové programy, prohlédněte si následující témata:

- "Konfigurace správce front pro přijetí připojení klienta na platformě Multiplatforms" na stránce 1034
- "Příprava a spuštění ukázkových programů na systému IBM i" na stránce 1036

- “Příprava a spuštění ukázkových programů na systému AIX and Linux” na stránce 1037
- “Příprava a spuštění ukázkových programů na systému Windows” na stránce 1038

Multi *Konfigurace správce front pro přijetí připojení klienta na platformě Multiplatforms*

Než budete moci spustit ukázkové aplikace, musíte nejprve vytvořit správce front. Poté můžete nakonfigurovat správce front tak, aby bezpečně přijímal příchozí požadavky na připojení od aplikací spuštěných v režimu klienta.

Než začnete

Ujistěte se, že správce front již existuje a byl spuštěn. Zadáním příkazu MQSC určete, zda jsou záznamy ověřování kanálu již povoleny:

```
DISPLAY QMGR CHLAUTH
```

Důležité: Tato úloha očekává, že budou povoleny záznamy ověřování kanálu. Pokud se jedná o správce front používaného jinými uživateli a aplikacemi, změna tohoto nastavení ovlivní všechny ostatní uživatele a aplikace. Pokud váš správce front nepoužívá záznamy ověřování kanálu, lze krok 4 nahradit alternativní metodou ověřování (například uživatelskou procedurou pro zabezpečení zprávy), která nastaví uživatele MCAUSER na *id-neprivilegovaného-uživatele* , který získáte v kroku “1” na stránce 1034.

Musíte vědět, který název kanálu vaše aplikace očekává, že jej bude používat, aby aplikace mohla kanál používat. Musíte také vědět, které objekty, například fronty nebo témata, které vaše aplikace očekává, aby je mohla používat.

Informace o této úloze

Tato úloha vytvoří neprivilegované ID uživatele, které má být použito pro klientskou aplikaci, která se připojuje ke správci front. Klientská aplikace má přístup pouze k tomu, aby mohla používat kanál, který potřebuje, a frontu, kterou potřebuje, pomocí tohoto ID uživatele.

Postup

1. Získejte ID uživatele v systému, ve kterém je spuštěn váš správce front. Pro tuto úlohu nesmí být toto ID uživatele privilegovaným administrativním uživatelem. Toto ID uživatele bude oprávnění, pod kterým bude spuštěno připojení klienta ve správci front.
2. Spusťte program listener pomocí následujících příkazů, kde:

   qmgr-name je název správce front.
   nnnn je vámi zvolené číslo portu

   a) ALW
   Pro systémy AIX, Linux, and Windows :

   ```
   runmqlsr -t tcp -m qmgr-name -p nnnn
   ```

   b) IBM i
   Pro IBM i:

   ```
   STRMQMLSR MQMNAME(qmgr-name) PORT(nnnn)
   ```

3. Pokud vaše aplikace používá systém SYSTEM.DEF.SVRCONN pak je tento kanál již definován. Pokud vaše aplikace používá jiný kanál, vytvořte jej zadáním následujícího příkazu MQSC:

   ```
   DEFINE CHANNEL(' channel-name ') CHLTYPE(SVRCONN) TRPTYPE(TCP) +
   DESCR('Channel for use by sample programs')
   ```

   kde *název-kanálu* je název vašeho kanálu.

4. Vytvořte pravidlo ověřování kanálu, které umožní kanálu používat pouze adresu IP klientského systému, a to zadáním následujícího příkazu MQSC:

```
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. Pokud je ukázková klientská aplikace spuštěna na stejném počítači jako správce front, použijte adresu IP '127.0.0.1', pokud se vaše aplikace bude připojovat pomocí 'localhost'. Pokud se bude připojovat několik různých klientských počítačů, můžete místo jediné adresy IP použít vzor nebo rozsah. Podrobnosti viz Generické adresy IP .
   non-privileged-user-id je ID uživatele, které jste získali v kroku “1” na stránce 1034
5. Pokud vaše aplikace používá systém SYSTEM.DEFAULT.LOCAL.QUEUE , pak je tato fronta již definována. Pokud vaše aplikace používá jinou frontu, vytvořte ji zadáním následujícího příkazu MQSC:

```
DEFINE QLOCAL(' queue-name ') DESCR('Queue for use by sample programs')
```

   kde *název-fronty* je název vaší fronty.
6. Udělte přístup pro připojení a dotazování správce front zadáním následujícího příkazu MQSC:

```
SET AUTHREC OBJTYPE(QMGR) PRINCIPAL(' non-privileged-user-id ') +
AUTHADD(CONNECT, INQ)
```

   kde *non-privileged-user-id* je ID uživatele, které jste získali v kroku “1” na stránce 1034 .
7. Jedná-li se o aplikaci typu point-to-point, tj. o aplikaci využívající fronty, udělte přístup umožňující dotazování a vkládání a získávání zpráv pomocí vaší fronty pomocí ID uživatele, které má být použito, zadáním následujících příkazů MQSC:

```
SET AUTHREC PROFILE(' queue-name ') OBJTYPE(QUEUE) +
PRINCIPAL(' non-privileged-user-id ') AUTHADD(PUT, GET, INQ, BROWSE)
```

   kde:

   název-fronty je název vaší fronty.
   non-privileged-user-id je ID uživatele, které jste získali v kroku “1” na stránce 1034
8. Pokud je vaše aplikace aplikací typu publikování/odběr, tj. používá témata, udělte přístup pro povolení publikování a přihlášení k odběru pomocí vašeho tématu pomocí ID uživatele, které má být použito, zadáním příkazů MQSC:

```
SET AUTHREC PROFILE('SYSTEM.BASE.TOPIC') OBJTYPE(TOPIC) +
PRINCIPAL(' non-privileged-user-id ') AUTHADD(PUB, SUB)
```

   kde:

   non-privileged-user-id je ID uživatele, které jste získali v kroku “1” na stránce 1034
   To umožní *non-privileged-user-id* přístup k libovolnému tématu ve stromu témat, případně můžete definovat objekt tématu pomocí **DEFINE TOPIC** a udělit přístup pouze k části stromu témat, na kterou odkazuje daný objekt tématu. Podrobnosti viz Řízení uživatelského přístupu k tématům .

Jak pokračovat dále

Aplikace klienta se nyní může připojit ke správci front a vkládat nebo získávat zprávy pomocí fronty.

Související pojmy

ALW Udělení přístupu k objektu IBM MQ na AIX, Linux, and Windows

Související odkazy
NASTAVIT CHLAUTH
Definovat kanál
DEFINICE QLOCAL
SET AUTHREC
IBM i IBM MQ oprávnění na IBM i

IBM i *Příprava a spuštění ukázkových programů na systému IBM i*
Před spuštěním ukázkových programů v systému IBM imusíte nejprve vytvořit správce front a také fronty, které potřebujete. Chcete-li spustit ukázky v jazyce COBOL, možná budete muset provést další přípravu.

Informace o této úloze

Zdroj pro ukázkové programy IBM MQ for IBM i je poskytován v knihovně QMQMSAMP jako členové 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 jej zkompilovat pomocí příkazu CRTCLPGM.

Chcete-li spustit ukázky, použijte buď spustitelné verze jazyka C, které jsou dodávány v knihovně QMQM, nebo je zkompilujte podobným způsobem jako jiné aplikace IBM MQ .

Postup

1. Vytvořte správce front a nastavte výchozí definice.

   To musíte provést dříve, než budete moci spustit některý z ukázkových programů. Další informace o vytváření správce front naleznete v tématu Administrace IBM MQ. Informace o konfiguraci správce front tak, aby bezpečně přijímal příchozí požadavky na připojení z aplikací spuštěných v režimu klienta, naleznete v tématu “Konfigurace správce front pro přijetí připojení klienta na platformě Multiplatforms” na stránce 1034.
2. Chcete-li volat jeden z ukázkových programů pomocí dat ze členu PUT v souboru AMQSDATA knihovny QMQMSAMP, použijte příkaz jako:

   ```
   CALL PGM(QMQM/AMQSPUT4) PARM('QMQMSAMP/AMQSDATA(PUT)')
   ```

   Poznámka: Aby mohl kompilovaný modul používat systém souborů IFS, uveďte volbu SYSIFCOPT (*IFSIO) na CRTCMOD, pak název souboru, předaný jako parametr, musí být uveden v následujícím formátu:

   ```
   home/me/myfile
   ```
3. Chcete-li použít verze COBOL příkladů Inquire, Set a Echo, změňte definice procesu před spuštěním těchto ukázek.

   Pro příklady Inquire, Set a Echo spustí definice ukázek verze C těchto ukázek. Chcete-li verze COBOL, musíte změnit definice procesu:

   - SYSTEM.SAMPLE.INQPROCESS
   - SYSTEM.SAMPLE.SETPROCESS
   - SYSTEM.SAMPLE.ECHOPROCESS

   V systému IBM imůžete použít příkaz **CHGMQMPRC** (podrobnosti viz Změna MQ Process (CHGMQMPRC)) nebo můžete příkaz **AMQSAMP4** upravit a spustit s alternativní definicí.
4. Spusťte ukázkové programy.

   Další informace o parametrech, které každá z ukázek očekává, naleznete v popisech jednotlivých ukázek.

Poznámka: V případě ukázkových programů v jazyce COBOL, když předáváte názvy front jako parametry, musíte zadat 48 znaků, 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 1031
Techniky demonstrované ukázkovými programy pro systémy IBM MQ na systému IBM i .

Příprava a spuštění ukázkových programů na systému AIX and Linux

Před spuštěním ukázkových programů v systému AIX and Linuxmusíte nejprve vytvořit správce front a také fronty, které potřebujete. Chcete-li spustit ukázky v jazyce COBOL, možná budete muset provést další přípravu.

Informace o této úloze

Ukázkové soubory IBM MQ na systémech AIX and Linux jsou v adresářích uvedených v seznamu Tabulka 160 na stránce 1037 , pokud byly při instalaci použity výchozí hodnoty.

Tabulka 160. Kde najít ukázky pro IBM MQ na systémech AIX 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 budou pracovat. Sadu můžete vytvořit buď pomocí vlastních front, nebo spuštěním ukázkového souboru MQSC amqscos0.tst . Chcete-li spustit ukázky, použijte buď dodané spustitelné verze, nebo kompilujte zdrojové verze jako jiné aplikace pomocí kompilátoru ANSI.

Postup

1. Vytvořte správce front a nastavte výchozí definice.

   To musíte provést dříve, než budete moci spustit některý z ukázkových programů. Další informace o vytváření správce front naleznete v tématu Administrace IBM MQ. Informace o konfiguraci správce front tak, aby bezpečně přijímal příchozí požadavky na připojení z aplikací spuštěných v režimu klienta, naleznete v tématu “Konfigurace správce front pro přijetí připojení klienta na platformě Multiplatforms” na stránce 1034.
2. Pokud nepoužíváte vlastní fronty, spusťte ukázkový soubor MQSC amqscos0.tst a vytvořte sadu front.

   Chcete-li to provést na systémech AIX and Linux , zadejte:

   ```
   runmqsc QManagerName <amqscos0.tst > /tmp/sampobj.out
   ```

   Zkontrolujte soubor `sampobj.out` a ujistěte se, že nedošlo k žádným chybám.
3. Chcete-li použít verze COBOL příkladů Inquire, Set a Echo, změňte definice procesu před spuštěním těchto ukázek.

   Pro příklady Inquire, Set a Echo spustí definice ukázek verze C těchto ukázek. Chcete-li verze COBOL, musíte změnit definice procesu:

   - SYSTEM.SAMPLE.INQPROCESS
   - SYSTEM.SAMPLE.SETPROCESS
   - SYSTEM.SAMPLE.ECHOPROCESS

V systému Windowsto proveďte úpravou souboru `amqscos0.tst` a změnou názvů spustitelných souborů C na názvy spustitelných souborů COBOL před použitím příkazu **runmqsc** ke spuštění těchto ukázek.

4. Spusťte ukázkové programy.

   Chcete-li spustit ukázku, zadejte její název následovaný libovolnými parametry, například:

   ```
   amqsput myqueue qmanagername
   ```

   kde *myqueue* je název fronty, do které budou zprávy vloženy, a *qmanagername* je správce front, který vlastní *myqueue*.

   Další informace o parametrech, které každá z ukázek očekává, naleznete v popisech jednotlivých ukázek.

   Poznámka: V případě ukázkových programů v jazyce COBOL, když předáváte názvy front jako parametry, musíte zadat 48 znaků, v případě potřeby vyplňující prázdné znaky. Cokoli jiného než 48 znaků způsobí, že program selže s kódem příčiny 2085.

Související odkazy

"Ukázky pro systémy AIX and Linux" na stránce 1025
Techniky demonstrované ukázkovými programy pro IBM MQ for AIX or Linux.

Windows *Příprava a spuštění ukázkových programů na systému Windows*

Před spuštěním ukázkových programů v systému Windowsmusíte nejprve vytvořit správce front a také fronty, které potřebujete. Chcete-li spustit ukázky v jazyce COBOL, možná budete muset provést další přípravu.

Informace o této úloze

Ukázkové soubory IBM MQ for Windows jsou v adresářích uvedených v části Tabulka 161 na stránce 1038, pokud byly při instalaci použity výchozí hodnoty. Instalační jednotka má výchozí hodnotu < c: >.

Tabulka 161. Kde najít ukázky pro IBM MQ for Windows

Obsah	**Adresář**
Zdrojový kód jazyka C	*MQ_INSTALLATION_PATH*\Tools\C\Samples-Ukázky
Zdrojový kód pro ukázku obslužné rutiny nedoručených dopisů	*MQ_INSTALLATION_PATH*\Tools\C\Samples\DLQ
Zdrojový kód v jazyce COBOL	*MQ_INSTALLATION_PATH*\Tools\Cobol \ Ukázky
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 \ Ukázky

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

Poznámka: Pro některé ukázky spustitelných souborů C jsou k dispozici 64bitové verze.

Ukázky potřebují sadu front, se kterými budou pracovat. Můžete buď použít vlastní fronty, nebo spustit ukázkový soubor MQSC amqscos0.tst a vytvořit sadu front. Chcete-li spustit ukázky, použijte buď dodané spustitelné verze, nebo zkompilujte zdrojové verze jako jiné aplikace IBM MQ for Windows .

Postup

1. Vytvořte správce front a nastavte výchozí definice.

   To musíte provést dříve, než budete moci spustit některý z ukázkových programů. Další informace o vytváření správce front naleznete v tématu Administrace IBM MQ. Informace o konfiguraci správce front tak, aby bezpečně přijímal příchozí požadavky na připojení z aplikací spuštěných v režimu klienta, naleznete v tématu “Konfigurace správce front pro přijetí připojení klienta na platformě Multiplatforms” na stránce 1034.
2. Pokud nepoužíváte vlastní fronty, spusťte ukázkový soubor MQSC amqscos0.tst a vytvořte sadu front.

   Chcete-li to provést na systémech Windows , zadejte:

   ```
   runmqsc QManagerName < amqscos0.tst > sampobj.out
   ```

   Zkontrolujte soubor `sampobj.out` a ujistěte se, že nedošlo k žádným chybám. Tento soubor je ve vašem aktuálním adresáři.
3. Chcete-li použít verze COBOL příkladů Inquire, Set a Echo, změňte definice procesu před spuštěním těchto ukázek.

   Pro příklady Inquire, Set a Echo spustí definice ukázek verze C těchto ukázek. Chcete-li verze COBOL, musíte změnit definice procesu:

   - SYSTEM.SAMPLE.INQPROCESS
   - SYSTEM.SAMPLE.SETPROCESS
   - SYSTEM.SAMPLE.ECHOPROCESS

   V systému Windowsto proveďte úpravou souboru `amqscos0.tst` a změnou názvů spustitelných souborů C na názvy spustitelných souborů COBOL před použitím příkazu **runmqsc** ke spuštění těchto ukázek.
4. Spusťte ukázkové programy.

   Chcete-li spustit ukázku, zadejte její název následovaný libovolnými parametry, například:

   ```
   amqsput myqueue qmanagername
   ```

   kde *myqueue* je název fronty, do které budou zprávy vloženy, a *qmanagername* je správce front, který vlastní *myqueue*.

   Další informace o parametrech, které každá z ukázek očekává, naleznete v popisech jednotlivých ukázek.

   Poznámka: V případě ukázkových programů v jazyce COBOL, když předáváte názvy front jako parametry, musíte zadat 48 znaků, 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 1028
Techniky demonstrované ukázkovými programy pro IBM MQ for Windows.

“Ukázky produktu Visual Basic pro produkt IBM MQ for Windows” na stránce 1031
Techniky demonstrované ukázkovými programy pro systémy IBM MQ na systému Windows .

Ukázkový program uživatelské procedury rozhraní API

Ukázková uživatelská procedura rozhraní API generuje trasování MQI do souboru určeného uživatelem s předponou definovanou v proměnné prostředí **MQAPI_TRACE_LOGFILE** .

Další informace o uživatelských rozhraních rozhraní API viz “Zápis a kompilace uživatelských procedur rozhraní API na platformě Multiplatforms” na stránce 920.

Zdroj
amqsaxe0.c

Binární
amqsaxe

Konfigurace pro ukázkovou uživatelskou proceduru

1. Přidejte následující informace do ApiExitLokální sekce souboru `qm.ini` .

   Jiné platformy 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. Nastavte proměnnou prostředí **MQAPI_TRACE_LOGFILE** .

   ```
   MQAPI_TRACE_LOGFILE=/tmp/MqiTrace
   ```

3. Spusťte svou aplikaci.

   Výstupní soubory jsou vytvořeny v adresáři `/tmp` s názvy jako: `MqiTrace.pid.tid.log`.

Ukázkový program Asynchronní spotřeba

Ukázkový program amqscbf demonstruje použití MQCB a MQCTL k asynchronnímu přijímání zpráv z více front.

amqscbf je poskytován jako zdrojový kód C a binární spustitelný soubor klienta a serveru na platformách AIX, Linux, and Windows .

Program je spuštěn z příkazového řádku a má 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
```

Zadejte více než jeden název fronty pro čtení zpráv z více front (ukázka podporuje maximálně deset front).

Poznámka: Parametr **Reconnect type** je platný pouze pro klientské programy.

Příklad

Příklad zobrazuje příkaz amqscbf spuštěný jako serverový program, který čte jednu zprávu z `QL1` a poté se zastavuje.

K vložení testovací zprávy do systému `QL1`použijte Průzkumníka systému IBM MQ . 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 ukazuje amqscbf

Ukázka ukazuje, jak číst zprávy z více front v pořadí jejich příchodu. To by vyžadovalo mnohem více kódu pomocí synchronního příkazu MQGET. V případě asynchronní spotřeby se nepožaduje systém výzev a správu podprocesů a úložišť provádí produkt IBM MQ. Příklad "reálného světa" by se musel vypořádat s chybami; v ukázkových chybách jsou vypsány na konzoli.

Vzorový kód má následující kroky:

1. definovat funkci zpětného volání spotřeby jednotlivých 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 každou z nich 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` nemusí být nastaveno pro každou frontu; jedná se o pole pouze pro vstup. Ke každé frontě však můžete přidružit jinou funkci zpětného volání.

4. Zahájit spotřebu zpráv,

```
MQCTL(Hcon,MQOP_START,&ctlo,&CompCode,&Reason);
```

5. Počkejte, až uživatel stiskne Enter a pak zastavte spotřebu zpráv,

```
MQCTL(Hcon,MQOP_STOP,&ctlo,&CompCode,&Reason);
```

6. Nakonec se odpojte od správce front,

```
MQDISC(&Hcon,&CompCode,&Reason);
```

Ukázkový program Asynchronní vložení

Zde se dozvíte o spuštění ukázky amqsapt a návrhu ukázkového programu Asynchronous Put.

Ukázkový program asynchronního vkládání vkládá zprávy do fronty pomocí asynchronního volání MQPUT a poté načítá informace o stavu pomocí volání MQSTAT. Název tohoto programu na různých platformách viz “Funkce demonstrované v ukázkových programech na platformě Multiplatforms” na stránce 1025 .

Spuštění ukázky amqsapt

Tento program má až 6 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 určen správce front, připojí se program amqsapt k výchozímu správci front.

Návrh ukázkového programu Asynchronous Put

Program používá volání MQOPEN se zadanými výstupními volbami nebo s volbami MQOO_OUTPUT a MQOO_FAIL_IF_QUIESCING k otevření cílové fronty pro vkládání zpráv.

Pokud nemůže otevřít frontu, program zobrazí chybovou zprávu obsahující kód příčiny vrácený voláním MQOPEN. Chcete-li zachovat jednoduchost programu, při tomto a následných voláních MQI použije program pro mnoho voleb výchozí hodnoty.

Pro každý řádek vstupu program načte text do vyrovnávací paměti a pomocí volání MQPUT s parametrem MQPMO_ASYNC_RESPONSE vytvoří zprávu datagramu obsahující text tohoto řádku a asynchronně ji vloží do cílové fronty. Program pokračuje, dokud nedosáhne konce vstupu nebo dokud neselže volání MQPUT. Pokud program dosáhne konce vstupu, uzavře frontu pomocí volání MQCLOSE.

Program poté vydá volání MQSTAT, vrátí strukturu MQSTS a zobrazí zprávy obsahující počet úspěšně vložených zpráv, počet vložených zpráv s varováním a počet selhání.

Ukázkové programy Procházet

Ukázkové programy Procházet procházejí zprávy ve frontě pomocí volání MQGET.

Názvy těchto programů viz “Funkce demonstrované v ukázkových programech na platformě Multiplatforms” na stránce 1025 .

Návrh ukázkového programu Procházet

Program otevře cílovou frontu pomocí volání MQOPEN s volbou MQOO_BROWSE. Pokud nemůže otevřít frontu, program zobrazí chybovou zprávu obsahující kód příčiny vrácený voláním MQOPEN.

Pro každou zprávu ve frontě program použije volání MQGET ke zkopírování zprávy z fronty a poté zobrazí data obsažená ve zprávě. Volání MQGET používá tyto volby:

MQGMO_BROWSE_NEXT
Po volání MQOPEN je kurzor procházení umístěn logicky před první zprávou ve frontě, takže tato volba způsobí vrácení ***první*** zprávy při prvním volání.

MQGMO_NO_WAIT
Program nečeká, pokud ve frontě nejsou žádné zprávy.

MQGMO_ACCEPT_TRUNCATED_MSG
Volání MQGET určuje vyrovnávací paměť pevné velikosti. Pokud je zpráva delší než tato vyrovnávací paměť, program zobrazí zkrácenou zprávu spolu s varováním, že zpráva byla zkrácena.

Program demonstruje, jak musíte vymazat pole *MsgId* a *CorrelId* struktury MQMD po každém volání MQGET, protože volání nastaví tato pole na hodnoty obsažené ve zprávě, kterou načte. Vymazání těchto polí znamená, že následná volání MQGET načítají zprávy v pořadí, ve kterém jsou zprávy zadrženy ve frontě.

Program pokračuje na konec fronty; volání MQGET vrátí kód příčiny MQRC_NO_MSG_AVAILABLE a program zobrazí varovnou zprávu. Pokud volání MQGET selže, program zobrazí chybovou zprávu, která obsahuje kód příčiny.

Program poté uzavře frontu pomocí volání MQCLOSE.

Ukázkové programy Procházet pro AIX, Linux, and Windows
Při procházení ukázkových programů v systému AIX, Linux, and Windowszvažte použití tohoto tématu.

Verze C programu má 2 parametry

1. Název zdrojové fronty (nezbytné)
2. Název správce front (volitelné)

Není-li určen správce front, 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 budou zprávy zobrazeny, a `qmanagername` je správce front, který vlastní `myqueue`.

Pokud při spuštění ukázky C vynecháte `qmanagername`, předpokládá se, že frontu vlastní výchozí správce front.

Verze COBOL nemá žádné parametry. Připojí se k výchozímu správci front a po jeho spuštění budete vyzváni:

```
Please enter the name of the target queue
```

Zobrazí se pouze prvních 50 znaků každé zprávy, následovaných znaky `- - - truncated` , pokud se jedná o tento případ.

Ukázkové programy Procházet na systému IBM i
Každý program načte kopie všech zpráv ve frontě, kterou zadáte při volání programu; zprávy zůstanou ve frontě.

Můžete použít dodanou frontu SYSTEM.SAMPLE.LOCAL; nejprve spusťte ukázkový program Put, abyste vložili některé zprávy do fronty. Můžete použít frontu SYSTEM.SAMPLE.ALIAS, což je název aliasu pro stejnou lokální frontu. Program pokračuje, dokud nedosáhne konce fronty nebo dokud neselže volání MQI.

Ukázky C umožňují zadat název správce front, obecně jako druhý parametr, podobně jako ukázky systémů Windows . Příklad:

```
CALL PGM(QMQM/AMQSTRG4) PARM('SYSTEM.SAMPLE.TRIGGER' 'QM01')
```

Není-li určen správce front, připojí se k výchozímu správci front. To je také relevantní pro ukázky RPG. V případě ukázek RPG však musíte zadat název správce front, a nepovolit tak jeho výchozí nastavení.

ALW *Ukázkový program prohlížeče*

Ukázkový program prohlížeče čte a zapisuje jak deskriptor zprávy, tak pole obsahu všech zpráv ve frontě.

Ukázkový program je napsán jako nástroj, nejen pro demonstraci techniky. Názvy těchto programů viz “Funkce demonstrované v ukázkových programech na platformě Multiplatforms” na stránce 1025 .

Tento program má tyto poziční parametry:

1. Název zdrojové fronty (povinné)
2. Název správce front (povinné)
3. Volitelný parametr pro vlastnosti (volitelný)

Pomocí následujících proměnných prostředí zadejte pověření, která se používají k ověření se správcem front:

MQSAMP_USER_ID
Nastavte na ID uživatele, které se má použít pro ověření připojení, pokud chcete použít ID uživatele a heslo pro ověření u správce front. Program vyzve k zadání hesla, které má doprovázet ID uživatele.

Linux V 9.3.4 AIX **MQSAMP_TOKEN**

Nastavte na neprázdnou hodnotu, chcete-li zadat token ověření pro ověření u správce front. Program vyzve k zadání tokenu ověření. Tokeny ověřování mohou být použity pouze ukázkou **amqsbcgc** , která používá vazby klienta.

Chcete-li spustit tyto programy, zadejte jeden z následujících příkazů:

- `amqsbcg` *`myqueue qmanagername`*
- `amqsbcgc` *`myqueue qmanagername`*

kde *`myqueue`* je název fronty, ve které mají být zprávy procházeny, a *`qmanagername`* je správce front, který vlastní *`myqueue`*.

Přečte každou zprávu z fronty a do stdout zapíše následující:

- Formátovaná pole deskriptoru zpráv
- Data zprávy (vypsaná v hexadecimálním a, kde je to možné, znakovém formátu)

Tabulka 162. Přípustné hodnoty pro parametr vlastnosti

Hodnota	Chování
0	Výchozí chování. Vlastnosti, které jsou doručeny aplikaci, závisí na atributu fronty **`PropertyControl`** , ze kterého je zpráva načtena.

1	Vytvoří se popisovač zprávy a použije se s MQGET. Vlastnosti zprávy s výjimkou těch, které jsou obsaženy v deskriptoru zprávy (nebo rozšíření), jsou zobrazeny podobným způsobem jako deskriptor zprávy. Příklad:
	`****Message properties****`
	`  property name: property value`
	Nebo pokud nejsou k dispozici žádné vlastnosti:
	`****Message properties****`
	`  None`
	Číselné hodnoty jsou zobrazeny pomocí `printf`, řetězcové hodnoty jsou uzavřeny v apostrofech a bajtové řetězce jsou uzavřeny do X a apostrofů, stejně jako pro deskriptor zprávy.

2	Je zadána volba MQGMO_NO_PROPERTIES, takže budou vráceny pouze vlastnosti deskriptoru zpráv.
3	Je zadána volba MQGMO_PROPERTIES_FORCE_MQRFH2 , aby byly v datech zprávy vráceny všechny vlastnosti.
4	Je zadána volba MQGMO_PROPERTIES_COMPATIBILITY, aby bylo možné vrátit všechny vlastnosti v závislosti na tom, zda je vlastnost IBM MQ 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 z tohoto obslužného programu naleznete v tématu Procházení front.

Ukázka transakce CICS

K dispozici je ukázkový transakční program CICS s názvem amqscic0.ccs pro zdrojový kód a amqscic0 pro spustitelnou verzi. Transakce můžete sestavit pomocí standardních zařízení CICS .

Podrobné informace o příkazech potřebných pro vaši platformu naleznete v části “Sestavení procedurální aplikace” na stránce 967 .

Transakce čte zprávy z přenosové fronty SYSTEM.SAMPLE.CICS.WORKQUEUE ve výchozím správci front a umístí je do lokální fronty, jejíž název je obsažen v záhlaví přenosu zprávy. Všechna selhání jsou odeslána do fronty SYSTEM.SAMPLE.CICS.DLQ.

Poznámka: K vytvoření těchto front a ukázkových vstupních front můžete použít ukázkový skript MQSC amqscic0.tst .

Ukázkový program Connect

Ukázkový program Connect vám umožňuje prozkoumat volání MQCONNX a jeho volby z klienta. Ukázka se připojuje ke správci front pomocí volání MQCONNX, zjišťuje název správce front pomocí volání MQINQ a zobrazuje jej. Také se dozvíte o spuštění ukázky amqscnxc.

Poznámka: Ukázkový program Connect je ukázkou klienta. Můžete jej zkompilovat a spustit na serveru, ale funkce je smysluplná pouze na klientovi a jsou dodány pouze spustitelné soubory klienta.

Spuštění ukázky amqscnxc

Syntaxe příkazového řádku ukázkového programu Connect je:

```
amqscnxc [-x ConnName [-c SvrconnChannelName]] [-u User] [QMgrName]
```

Parametry jsou volitelné a jejich pořadí není důležité s výjimkou parametru QMgrName, který, je-li uveden, musí být poslední. Parametry jsou:

ConnName
Název připojení TCP/IP správce front serveru

Pokud nezadáte název připojení TCP/IP, bude MQCONNX vydán s volbou *ClientConnPtr* nastavenou na hodnotu NULL.

Název kanálu SvrconnChannel
Název kanálu připojení serveru

Pokud zadáte název připojení TCP/IP, ale ne kanál připojení serveru (zpětný chod není povolen), ukázka použije název SYSTEM.DEF.SVRCONN.

Uživatel
Jméno uživatele, které se má použít pro ověření připojení

Zadáte-li toto, program vyzve k zadání hesla, které bude doprovázet toto ID uživatele.

QMgrName
Název cílového správce front

Pokud nezadáte cílového správce front, ukázka se připojí k kterémkoli správci front, který naslouchá na daném názvu připojení TCP/IP.

Poznámka: Zadáte-li otazník jako jediný parametr nebo zadáte-li nesprávné parametry, zobrazí se zpráva s vysvětlením, jak program používat.

Spustíte-li ukázku bez voleb příkazového řádku, obsah proměnné prostředí MQSERVER se použije k určení informací o připojení. (V tomto příkladu je MQSERVER nastaven na `SYSTEM.DEF.SVRCONN/TCP/machine.site.company.com`.) Výstup se zobrazí takto:

```
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 název připojení TCP/IP a název kanálu připojení serveru, ale žádný název cílového správce front, postupujte takto:

```
amqscnxc -x machine.site.company.com -c SYSTEM.ADMIN.SVRCONN
```

Použije se výchozí název správce front a zobrazí se 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 název připojení TCP/IP a název cílového správce front, postupujte takto:

```
amqscnxc -x machine.site.company.com MACHINE
```

Výstup se zobrazí takto:

```
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 Data-Conversion

Ukázkový program pro převod dat je kostrou uživatelské procedury pro převod dat. Zde se dozvíte o návrhu ukázky převodu dat.

Názvy těchto programů viz “Funkce demonstrované v ukázkových programech na platformě Multiplatforms” na stránce 1025 .

Návrh vzorku pro převod dat

Každá uživatelská procedura pro převod dat převádí jeden pojmenovaný formát zprávy. Tato kostra je určena jako modul wrapper pro fragmenty kódu generované obslužným programem pro generování uživatelské procedury pro převod dat.

Obslužný program vytvoří jeden fragment kódu pro každou datovou strukturu; několik takových struktur tvoří formát, takže do této kostry se přidá několik fragmentů kódu, aby se vytvořila rutina pro převod dat celého formátu.

Program poté zkontroluje, zda je převod úspěšný nebo neúspěšný, a vrátí volajícímu požadované hodnoty.

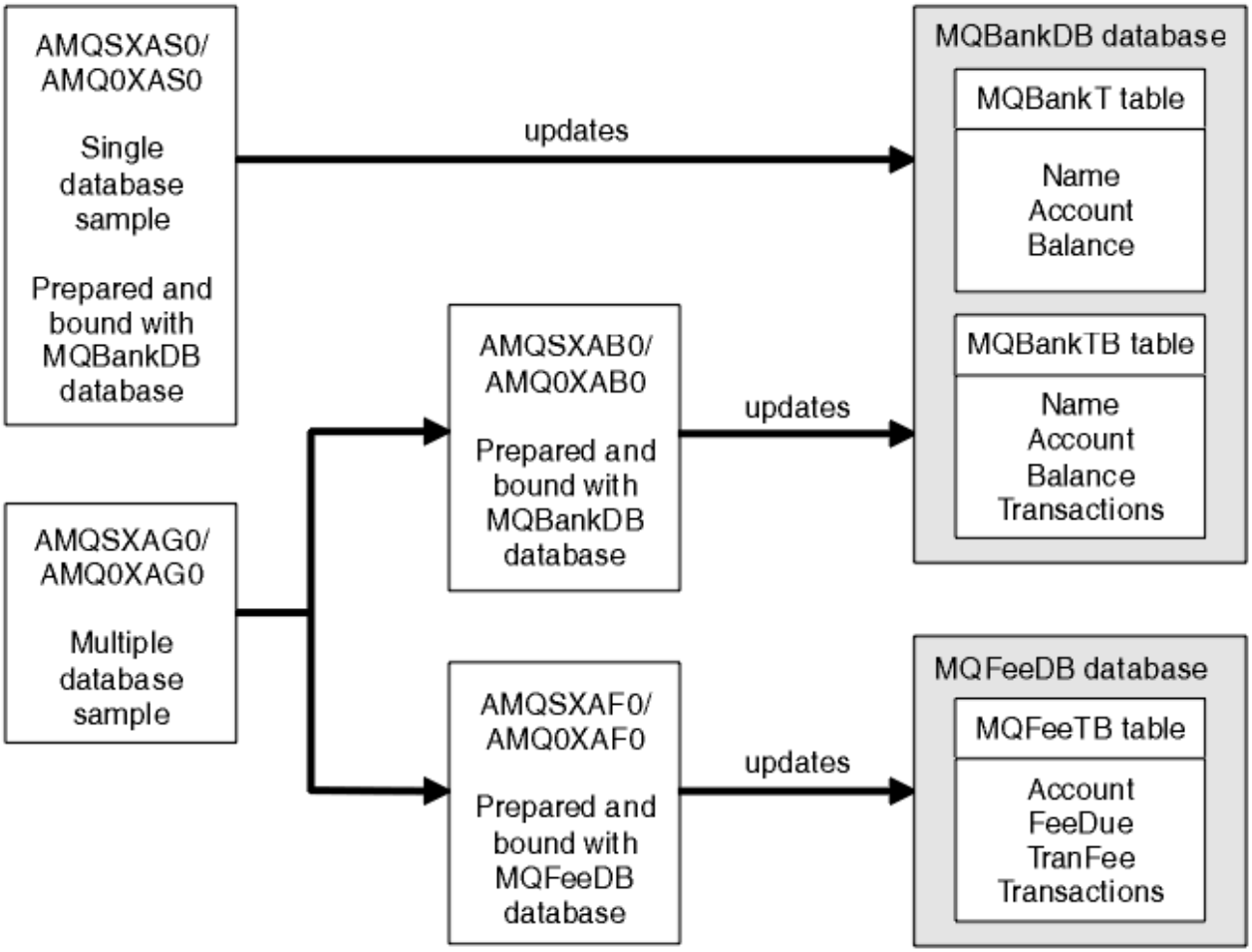
Ukázky koordinace databáze

K dispozici jsou dvě ukázky, které ukazují, jak může produkt IBM MQ koordinovat aktualizace produktu IBM MQ i aktualizace databáze v rámci stejné pracovní jednotky.

Jedná se o tyto ukázky:

1. AMQSXAS0 (v C) nebo AMQ0XAS0 (v COBOLu), který aktualizuje jednu databázi v rámci IBM MQ jednotky práce.
2. AMQSXAG0 (v C) nebo AMQ0XAG0 (v COBOLu), AMQSXAB0 (v C) nebo AMQ0XAB0 (v COBOLu) a AMQSXAF0 (v C) nebo AMQ0XAF0 (v COBOLu), které společně aktualizují dvě databáze v rámci pracovní jednotky IBM MQ a ukazují, jak lze přistupovat k více databázím. Tyto ukázky slouží k zobrazení použití volání MQBEGIN, smíšených volání SQL a IBM MQ a místa a místa připojení k databázi.

Obrázek 128 na stránce 1047 ukazuje, jak jsou poskytnuté ukázky použity k aktualizaci databází:



Obrázek 128. Ukázky koordinace databáze

Programy načtou zprávu z fronty (pod synchronizačním bodem) a poté pomocí informací ve zprávě získají příslušné informace z databáze a aktualizují je. Pak se vytiskne nový stav databáze.

Logika programu je následující:

1. Použít název vstupní fronty z argumentu programu
2. Připojit se k výchozímu správci front (nebo volitelně k zadanému názvu v jazyce C) pomocí MQCONN
3. Otevřít frontu (pomocí MQOPEN) pro vstup bez selhání
4. Spuštění pracovní jednotky pomocí příkazu 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 je k dispozici, se nepočítá jako selhání a smyčka končí)
10. Zavřít frontu pomocí příkazu MQCLOSE
11. Odpojení od fronty pomocí MQDISC

V ukázkách se používají kurzory SQL, takže čtení z databází (tj. více instancí) je uzamčeno během zpracování zprávy, což umožňuje souběžné spuštění více instancí těchto programů. Kurzory jsou explicitně otevřeny, ale implicitně zavřeny voláním MQCMIT.

Jedna ukázka databáze (AMQSXAS0 nebo AMQ0XAS0) nemá žádné příkazy SQL CONNECT a připojení k databázi je implicitně vytvořeno produktem IBM MQ pomocí volání 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 váš databázový produkt nebo pokud přistupujete k jedné databázi ve více databázových produktech, lze příkazy SQL CONNECT odebrat.

Ukázky jsou připraveny s databázovým produktem IBM Db2 , takže je možná budete muset upravit tak, aby spolupracovaly s dalšími databázovými produkty.

Kontrola chyb SQL používá rutiny v jazyce UTIL.C a CHECKERR.CBL dodává Db2. Před kompilací a propojením musí být tyto soubory zkompilovány nebo nahrazeny.

Poznámka: Používáte-li zdroj Micro Focus COBOL CHECKERR.MFC pro kontrolu chyb SQL, musíte změnit ID programu na velká písmena, tj. CHECKERR, aby se AMQ0XAS0 správně propojil.

Vytvoření databází a tabulek
Před kompilací ukázek vytvořte databáze a tabulky.

Chcete-li vytvořit databáze, použijte pro svůj databázový produkt obvyklou metodu, například:

```
DB2 CREATE DB MQBankDB
DB2 CREATE DB MQFeeDB
```

Vytvořte tabulky pomocí příkazů SQL takto:

V jazyce 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 COBOLu:

```
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 takto:

```
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 propojení ukázek
Získejte informace o předkompilaci, kompilaci a propojování ukázek v jazycích C a COBOL.

Předkompilujte soubory `.SQC` (v jazyce C) a soubory `.SQB` (v jazyku COBOL) a svažte je s příslušnou databází, abyste vytvořili soubory `.C` nebo `.CBL` . Chcete-li tak učinit, použijte typickou metodu databázového produktu.

Předkompilace v 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

Následující ukázkové příkazy používají symboly *DB2TOP* a *MQ_INSTALLATION_PATH*. *DB2TOP* představuje instalační adresář produktu Db2 . *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

- AIX V systému AIXje cesta k adresáři:

```
/usr/lpp/db2_05_00
```

- Windows Na systémech Windows závisí cesta k adresáři na cestě zvolené při instalaci produktu. Pokud jste zvolili výchozí nastavení, cesta je následující:

```
c:\sqllib
```

Poznámka: Před zadáním příkazu link na systémech Windows se ujistěte, že proměnná prostředí LIB obsahuje cesty ke 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:

  – Linux AIX Na systémech AIX and Linux:

  ```
  MQ_INSTALLATION_PATH/samp/xatm
  ```

  – Windows Na systémech Windows:

  ```
  MQ_INSTALLATION_PATH\tools\c\samples\xatm
  ```

- Soubory `.c` , které jste získali předkompilací zdrojových souborů `.sqc` , `amqsxas0.sqc`, `amqsxaf0.sqca` a `amqsxab0.sqc`.
- Soubory `util.c` a `util.h` z vaší instalace Db2 .

  Poznámka: Tyto soubory lze nalézt v adresáři:

  ```
  DB2TOP/samples/c
  ```

Sestavte soubory objektů pro každý soubor `.c` pomocí následujícího příkazu kompilátoru pro používanou platformu:

- AIX AIX

```
xlc_r -I MQ_INSTALLATION_PATH/inc -I DB2TOP/include -c -o
FILENAME.o FILENAME.c
```

- Windows Systémy Windows

```
cl /c /I MQ_INSTALLATION_PATH\tools\c\include /I DB2TOP\include
FILENAME.c
```

Sestavte spustitelný soubor `amqsxag0` pomocí následujícího příkazu odkazu pro používanou platformu:

- AIX 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
```

- Windows Systémy Windows

```
link util.obj amqsxaf0.obj amqsxab0.obj amqsxag0.obj mqm.lib db2api.lib
/out:amqsxag0.exe
```

Sestavte spustitelný soubor `amqsxas0` pomocí následujících kompilujících a propojovacích příkazů pro používanou platformu:

- AIX AIX

```
xlc_r -H512 -T512 -L DB2TOP/lib -ldb2
-L MQ_INSTALLATION_PATH/lib -lmqm util.o amqsxas0.o -o amqsxas0
```

- Windows Systémy Windows

```
link util.obj amqsxas0.obj mqm.lib db2api.lib /out:amqsxas0.exe
```

Další informace

AIX Pokud pracujete na serveru AIX a chcete přistupovat k databázi Oracle, použijte kompilátor xlc_r a odkaz na soubor libmqm_r.a.

Spuštění ukázek
Pomocí těchto informací se dozvíte, jak nakonfigurovat správce front před spuštěním ukázek koordinace databází v jazycích C a COBOL.

Před spuštěním ukázek konfigurujte správce front s databázovým produktem, který používáte. Informace o postupu naleznete v tématu Scénář 1: Koordinace provádí správce front.

Následující názvy poskytují informace o tom, jak spouštět ukázky v jazycích C a COBOL:

- "C ukázky" na stránce 1051
- "Ukázky v jazyce COBOL" na stránce 1052

C ukázky

Zprávy musí být v následujícím formátu, aby mohly být čteny z fronty:

```
UPDATE Balance change=nnn WHERE Account=nnn
```

AMQSPUT lze použít k vložení zpráv do fronty.

Ukázky koordinace databáze berou dva parametry:

1. Název fronty (povinné)
2. Název správce front (volitelné)

Za předpokladu, že jste vytvořili a nakonfigurovali správce front pro jednu ukázku databáze s názvem singDBQMs frontou s názvem singDBQ, zvýšíte účet pana Freda Bloggse o 50 takto:

```
AMQSPUT singDBQ singDBQM
```

Pak klíč v následující zprávě:

```
UPDATE Balance change=50 WHERE Account=1
```

Do fronty můžete vložit více zpráv.

```
AMQSXAS0 singDBQ singDBQM
```

Poté se vytiskne aktualizovaný stav účtu pana Freda Bloggse.

Za předpokladu, že jste vytvořili a nakonfigurovali správce front pro ukázku s více databázemi s názvem multDBQMs frontou s názvem multDBQ, snížíte účet paní Mary Brown o 75 následujícím způsobem:

```
AMQSPUT multDBQ multDBQM
```

Pak klíč v následující zprávě:

```
UPDATE Balance change=-75 WHERE Account=3
```

Do fronty můžete vložit více zpráv.

```
AMQSXAG0 multDBQ multDBQM
```

Poté se vytiskne aktualizovaný stav účtu paní Mary Brownové.

Ukázky v jazyce COBOL

Zprávy musí být v následujícím formátu, aby mohly být čteny z fronty:

```
UPDATE Balance change=snnnnnnnn WHERE Account=nnnnnnnn
```

Pro zjednodušení musí být `Balance change` osmiznakové číslo se znaménkem a `Account` musí být osmiznakové číslo.

K vložení zpráv do fronty lze použít ukázku AMQSPUT.

Ukázky neberou žádné parametry a používají výchozího správce front. Lze ji kdykoli nakonfigurovat tak, aby spustila pouze jednu z ukázek. Za předpokladu, že jste nakonfigurovali výchozího správce front pro jednu ukázku databáze s frontou s názvem singDBQ, zvýšíte účet pana Freda Bloggse o 50 takto:

```
AMQSPUT singDBQ
```

Pak klíč v následující zprávě:

```
UPDATE Balance change=+00000050 WHERE Account=00000001
```

Do fronty můžete vložit více zpráv:

```
AMQ0XAS0
```

Zadejte název fronty:

```
singDBQ
```

Poté se vytiskne aktualizovaný stav účtu pana Freda Bloggse.

Za předpokladu, že jste nakonfigurovali výchozího správce front pro ukázku více databází s frontou s názvem multDBQ, snížíte účet paní Mary Brown o 75 takto:

```
AMQSPUT multDBQ
```

Pak klíč v následující zprávě:

```
UPDATE Balance change=-00000075 WHERE Account=00000003
```

Do fronty můžete vložit více zpráv:

```
AMQ0XAG0
```

Zadejte název fronty:

```
multDBQ
```

Poté se vytiskne aktualizovaný stav účtu paní Mary Brownové.

Ukázka obslužné rutiny fronty nedoručených zpráv

Je poskytnuta ukázková obslužná rutina fronty nedoručených zpráv, název spustitelné verze je `amqsdlq`. Chcete-li obslužnou rutinu fronty nedoručených zpráv, která se liší od obslužné rutiny **RUNMQDLQ**, je k dispozici zdroj ukázky, který můžete použít jako základ.

Ukázka je podobná obslužné rutině nedoručených zpráv poskytnuté v rámci produktu, ale trasování a hlášení chyb se liší. Máte k dispozici dvě proměnné prostředí:

ODQ_TRACE
Nastavte na hodnotu YES nebo `yes` , chcete-li zapnout trasování.

ODQ_MSG
Nastavte na název souboru obsahujícího chybové a informační zprávy. Poskytnutý soubor se nazývá `amqsdlq.msg`.

Tyto proměnné musíte ve svém prostředí znát pomocí příkazů **`export`** nebo **`set`** , v závislosti na vaší platformě. Trasování je vypnuto pomocí příkazu **`unset`** .

Soubor chybových zpráv `amqsdlq.msg`můžete upravit tak, aby vyhovoval vašim vlastním požadavkům. Ukázka vkládá zprávy do souboru stdout, **ne** do souboru protokolu chyb IBM MQ .

Další informace o tom, jak obslužná rutina nedoručených zpráv funguje a jak ji spouštíte, naleznete v tématu Zpracování zpráv ve IBM MQ frontě nedoručených zpráv nebo v příručce *System Management Guide* pro vaši platformu.

Ukázkový program distribučního seznamu

Ukázka rozdělovníku amqsptl0 uvádí příklad vložení zprávy do několika front zpráv. Je založen na ukázce MQPUT amqsput0.

Spuštění ukázky Distribuční seznam, amqsptl0

Ukázka distribučního seznamu se spouští podobným způsobem jako ukázky Put.

Používá následující parametry:

- Názvy front
- Názvy správců front

Tyto hodnoty se zadávají jako dvojice. Příklad:

```
amqsptl0 queue1 qmanagername1 queue2 qmanagername2
```

Fronty se otevírají pomocí MQOPEN a zprávy se vkládají do front pomocí MQPUT. Pokud některý z názvů front nebo správců front není rozpoznán, jsou vráceny kódy příčiny.

Nezapomeňte definovat kanály mezi správci front tak, aby mezi nimi mohly proudit zprávy. Ukázkový program to pro vás nedělá.

Návrh vzorku distribučního seznamu

Záznamy vložených zpráv (MQPMR) určují atributy zpráv pro každé místo určení. Ukázka poskytuje hodnoty pro *`MsgId`* a *`CorrelId`*a tyto přepisují hodnoty určené ve struktuře MQMD.

Pole *`PutMsgRecFields`* ve struktuře MQPMO označuje, která pole jsou přítomna v MQPMR:

```
MQLONG PutMsgRecFields=MQPMRF_MSG_ID + MQPMRF_CORREL_ID;
```

Dále ukázka přidělí záznamy odezvy a záznamy objektů. Záznamy objektů (MQORs) vyžadují alespoň jednu dvojici názvů a sudý počet názvů, tj. *ObjectName* a *ObjectQMgrName*.

Další fáze zahrnuje připojení ke správcům front pomocí MQCONN. Ukázka se pokusí o připojení ke správci front přidruženému k první frontě v úložišti MQOR. Pokud se tato operace nezdaří, budou postupně procházet záznamy objektů. Budete informováni, pokud se nelze připojit k žádnému správci front a ukončení programu.

Cílové fronty jsou otevřeny pomocí MQOPEN a zpráva je vložena do těchto front pomocí MQPUT. Všechny problémy a selhání jsou nahlášeny v záznamech odezvy (MQRRs).

Nakonec jsou cílové fronty uzavřeny pomocí příkazu MQCLOSE a program se odpojí od správce front pomocí příkazu MQDISC. Stejné záznamy odpovědí se používají pro každé volání, které uvádí *CompCode* a *Reason*.

Ukázkové programy Echo

Ukázkové programy Echo ozvěte zprávu z fronty zpráv do fronty odpovědí.

Názvy těchto programů viz “Funkce demonstrované v ukázkových programech na platformě Multiplatforms” na stránce 1025 .

Programy jsou určeny ke spuštění jako spuštěné programy.

V systémech IBM i, AIX, 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 Aby v systému IBM ifungoval spouštěcí proces, ujistěte se, že ukázkový program Echo, který chcete použít, je spuštěn zprávami přicházejícími do fronty SYSTEM.SAMPLE.ECHO. Chcete-li to provést, zadejte název ukázkového programu Echo, který chcete použít, do pole *ApplId* definice procesu SYSTEM.SAMPLE.ECHOPROCESS. (Pro tento účel můžete použít příkaz CHGMQMPRC; podrobnosti viz Změnit proces MQ (CHGMQMPRC).) Ukázková fronta má typ spouštěče FIRST, takže pokud již existují zprávy ve frontě před spuštěním ukázky požadavku, není ukázka Echo spuštěna odesílanými zprávami.

Když jste nastavili definici správně, nejprve spusťte AMQSERV4 v jedné úloze, pak spusťte AMQSREQ4 v jiné úloze. Můžete použít AMQSTRG4 místo AMQSERV4, ale potenciální prodlevy při odesílání úloh by mohly způsobit, že bude méně snadné sledovat, co se děje.

Ukázkové programy požadavků použijte k odeslání zpráv do fronty SYSTEM.SAMPLE.ECHO. Ukázkové programy Echo odešlou zprávu odpovědi obsahující data ve zprávě požadavku do fronty pro odpověď uvedené ve zprávě požadavku.

Návrh vzorových programů Echo

Program otevře frontu uvedenou ve struktuře zprávy spouštěče, která byla předána při jejím spuštění. (Pro přehlednost se na tuto frontu odkazuje jako na frontu požadavků.) Program používá volání MQOPEN k otevření této fronty pro sdílený vstup.

Program používá volání MQGET k odebrání 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, aby zjistil, zda se jedná o zprávu požadavku; pokud není, program zprávu vyřadí a zobrazí varovnou zprávu.

Pro každý řádek vstupu pak program načte text do vyrovnávací paměti a použije volání MQPUT1 k vložení zprávy požadavku obsahující text tohoto řádku do fronty pro odpověď.

Pokud volání MQGET selže, program vloží zprávu sestavy do fronty pro odpověď a nastaví pole *Feedback* deskriptoru zprávy na kód příčiny vrácený příkazem MQGET.

Pokud ve frontě požadavků nezbývají 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 dodán žádný vzorek. Aby program ECHO fungoval:

- Napište program, správně uveďte parametry **Format**, **Encoding**a **CCSID** , abyste odeslali zprávy s textovým požadavkem.

  Program ECHO požádá správce front o provedení konverze dat zpráv, je-li to potřeba.
- Uveďte CONVERT (*YES) na odesílacím kanálu IBM MQ for IBM i , pokud program, který jste napsali, neposkytuje podobnou konverzi pro odpověď.

Ukázkové programy Get

Ukázkové programy Get získají zprávy z fronty pomocí volání MQGET.

Názvy těchto programů viz “Funkce demonstrované v ukázkových programech na platformě Multiplatforms” na stránce 1025 .

Návrh ukázkového programu Get

Program otevře cílovou frontu pomocí volání MQOPEN s volbou MQOO_INPUT_AS_Q_DEF. Pokud nemůže otevřít frontu, program zobrazí chybovou zprávu 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 s hodnotou *WaitInterval* 15 sekund, takže program čeká po tuto dobu, pokud ve frontě není žádná zpráva. Pokud před vypršením tohoto intervalu nepřijde žádná zpráva, volání selže a vrátí kód příčiny MQRC_NO_MSG_AVAILABLE.

Program demonstruje, jak musíte vymazat pole *MsgId* a *CorrelId* struktury MQMD po každém volání MQGET, protože volání nastaví tato pole na hodnoty obsažené ve zprávě, kterou načte. Vymazání těchto polí znamená, že následná volání MQGET načítají zprávy v pořadí, ve kterém jsou zprávy zadrženy ve frontě.

Volání MQGET určuje vyrovnávací paměť pevné velikosti. Pokud je zpráva delší než tato vyrovnávací paměť, volání selže a program se zastaví.

Program pokračuje, dokud volání MQGET nevrátí kód příčiny MQRC_NO_MSG_AVAILABLE nebo dokud volání MQGET neselže. Pokud volání selže, program zobrazí chybovou zprávu, která obsahuje kód příčiny.

Program poté uzavře frontu pomocí volání MQCLOSE.

Spuštění ukázek amqsget a amqsgetc

Každý z těchto programů má následující poziční parametry:

1. Název zdrojové fronty (povinné)
2. Název správce front (volitelné)

   Není-li určen správce front, produkt **amqsget** se připojí k výchozímu správci front a produkt **amqsgetc** se připojí ke správci front určenému proměnnou prostředí *MQSERVER* nebo souborem definice kanálu klienta.
3. Volby otevření (volitelné)

   Pokud nejsou zadány otevřené volby, ukázka použije hodnotu 8193, která je kombinací těchto dvou voleb:

   - MQOO_INPUT_AS_Q_DEF
   - MQOO_FAIL_IF_QUIESCING
4. Volby zavření (volitelné)

   Nejsou-li zadány volby zavření, ukázka použije hodnotu 0, která je MQCO_NONE.

Pomocí následujících proměnných prostředí zadejte pověření, která se používají k ověření se správcem front:

MQSAMP_USER_ID
Nastavte na ID uživatele, které se má použít pro ověření připojení, pokud chcete použít ID uživatele a heslo pro ověření u správce front. Program vyzve k zadání hesla, které má doprovázet ID uživatele.

Linux V 9.3.4 AIX **MQSAMP_TOKEN**
Nastavte na neprázdnou hodnotu, chcete-li zadat token ověření pro ověření u správce front. Program vyzve k zadání tokenu ověření. Tokeny ověřování mohou být použity pouze ukázkou **amqsgetc** , která používá vazby klienta.

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

Všimněte si, že produkt **amqsget** provádí lokální připojení ke správci front s použitím sdílené paměti pro připojení ke správci front a jako takový může být spuštěn 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 zadat podrobnosti aplikace o tom, jak se skutečně dostat ke správci front, pokud jde o hostitele správce front nebo adresu IP a port modulu listener správce front.

Obvykle se tak děje buď pomocí proměnné prostředí **MQSERVER** , nebo definováním podrobností o připojení pomocí tabulky definic kanálů klienta, kterou lze také poskytnout produktu **amqsgetc** pomocí proměnných prostředí; například viz MQCCDTURL.

Příklad použití produktu **MQSERVER**pro lokální připojení ke správci front, který má modul listener spuštěný na portu 1414 a používající výchozí kanál připojení serveru, je:

```
export MQSERVER="SYSTEM.DEF.SVRCONN/TCP/ localhost(1414)"
```

Ukázkové programy s vysokou dostupností

Ukázkové programy **amqsghac**, **amqsphac**a **amqsmhac** vysoké dostupnosti používají automatické opětovné připojení klienta k demonstraci zotavení po selhání správce front. Produkt **amqsfhac** kontroluje, zda správce front používající síťové úložiště udržuje integritu dat po selhání.

Programy **amqsghac**, **amqsphac**a **amqsmhac** jsou spuštěny 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 ke správcům front s jednou instancí, kteří jsou obvykle konfigurováni ve skupině správců front.

Chcete-li zachovat jednoduchý příklad, takže je snadné jej konfigurovat, zobrazí se ukázkové programy, které se znovu připojují k jednomu správci front instance, který je spuštěn, zastaven a poté znovu restartován; viz “Nastavení a řízení správce front” na stránce 1058.

Pomocí příkazu **amqsfhac** paralelně s příkazem **amqmfsck** zkontrolujte integritu systému souborů. Další informace viz **amqmfsck** (kontrola systému souborů) a Ověření chování sdíleného systému souborů .

amqsphac *queueName* [*qMgrNázev*]

- **amqsphac** je aplikace IBM MQ MQI client . Vloží posloupnost zpráv do fronty s dvousekundovou prodlevou mezi každou zprávu a zobrazí události odeslané do obslužné rutiny událostí.
- Pro vložení zpráv do fronty není použit žádný synchronizační bod.
- Opětovné připojení lze provést pro libovolného správce front ve stejné skupině správců front.

amqsghac *queueName* [*qMgrNázev*]

- **amqsghac** je aplikace IBM MQ MQI client . Získává zprávy z fronty a zobrazuje události odeslané do obslužné rutiny událostí.
- K získání zpráv z fronty není použit žádný synchronizační bod.
- Opětovné připojení lze provést pro libovolného správce front ve stejné skupině správců front.

amqsmhac -s *sourceQueueNázev* -t *targetQueueNázev* [-m *qMgrNázev*] [-w *waitInterval*]

- **amqsmhac** je aplikace IBM MQ MQI client . Kopíruje zprávy z jedné fronty do druhé s předvoleným intervalem čekání 15 minut po poslední zprávě, která je přijata před dokončením programu.
- Zprávy jsou kopírovány v synchronizačním bodu.
- Opětovné připojení lze vytvořit pouze pro stejného správce front.

amqsfhac *QueueManager QueueName SideQueue InTransactionPočet RepeatCount* (0 | 1 | 2)

- **amqsfhac** je aplikace IBM MQ MQI client . Kontroluje, zda správce front IBM MQ s více instancemi pomocí síťového úložiště, jako je NAS nebo klastrový systém souborů, udržuje integritu dat. Chcete-li spustit příkaz **amqsfhac** v části Ověření chování sdíleného systému souborů, postupujte takto.
- Při připojování k produktu *QueueManagerName*používá volbu `MQCNO_RECONNECT_Q_MGR` . Při selhání správce front se automaticky znovu připojí.
- Vkládá *InTransactionPočet***RepeatCount* trvalých zpráv do *QueueName* , během kterého dochází k selhání správce front v libovolném počtu případů. Produkt **amqsfhac** se pokaždé znovu připojí ke správci front a pokračuje. Testem je zajistit, aby nebyly ztraceny žádné zprávy.
- *InTransactionPočet* zpráv se vloží do každé transakce. Počet opakování transakce *RepeatCount* . Dojde-li v rámci transakce k selhání, produkt **amqsfhac** transakci odvolá a znovu odešle, když se produkt **amqsfhac** znovu připojí ke správci front.
- Také vkládá zprávy do *SideQueueNázev*. Používá *SideQueueName* ke kontrole, zda jsou všechny zprávy úspěšně potvrzeny nebo odvolány z *QueueName* . Pokud zjistí nekonzistenci, zapíše chybovou zprávu.
- Změňte velikost výstupního trasování z **amqsfhac** nastavením posledního parametru na `(0|1|2)`.

  0
  Nejmenší výstup.

  1
  Střední výstup.

  2
  Většina výstupu.

Konfigurace připojení klienta

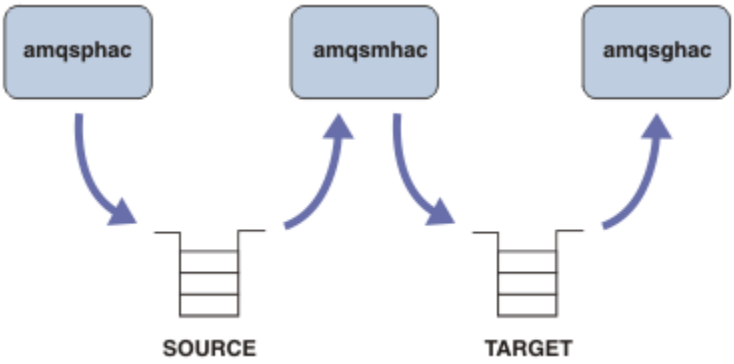
Chcete-li spustit ukázky, musíte nakonfigurovat kanál připojení klienta a serveru. Procedura verifikace klienta vysvětluje, jak nastavit testovací prostředí klienta.

Případně použijte konfiguraci uvedenou v následujícím příkladu.

Příklad použití amqsghac, amqsphaca amqsmhac

Příklad demonstruje klienty s možností opětovného připojení pomocí správce front s jednou instancí.

Zprávy jsou umístěny do fronty SOURCE pomocí **amqsphac**, přeneseny do TARGET pomocí **amqsmhac**a načteny z TARGET pomocí **amqsghac** ; viz Obrázek 129 na stránce 1058.



Obrázek 129. Znovu připojitelné ukázky klienta

Chcete-li spustit ukázky, postupujte takto.

1. Vytvořte soubor `hasamples.tst` obsahující příkazy:

```
DEFINE QLOCAL(SOURCE) REPLACE
DEFINE QLOCAL(TARGET) REPLACE
DEFINE CHANNEL(CHANNEL1) CHLTYPE(SVRCONN)  TRPTYPE(TCP) +
       MCAUSER(MUSR_MQADMIN) REPLACE
DEFINE CHANNEL(CHANNEL1) CHLTYPE(CLNTCONN) TRPTYPE(TCP) +
       CONNAME('LOCALHOST(2345)') QMNAME(QM1) REPLACE
ALTER  LISTENER(SYSTEM.DEFAULT.LISTENER.TCP) TRPTYPE(TCP) +
       PORT(2345)
START  LISTENER(SYSTEM.DEFAULT.LISTENER.TCP)
START  CHANNEL(CHANNEL1)
```

2. 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 souboru definice kanálu klienta `AMQCLCHL.TAB` , například `SET MQCHLLIB=C:\IBM\MQ\MQ7\Data\qmgrs\QM1\@ipcc`.
4. Otevřete tři nová okna se sadou **MQCHLLIB** ; například v systému Windowszadejte **start** třikrát na předchozí příkazový řádek a spusťte každý program v jednom z oken. Viz krok “5” na stránce 1059 v části “Nastavení a řízení správce front” na stránce 1058.)
5. Zadáním příkazu `endmqm -r -p QM1` zastavte správce front a poté umožněte klientům opětovné připojení.
6. Zadáním příkazu `strmqm QM1` restartujte správce front.

Výsledky spuštění ukázek **amqsghac**, **amqsphac**a **amqsmhac** na systému Windows jsou uvedeny 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 nastavili proměnnou **MQCHLLIB** později.

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.

```
C:\> runmqsc QM1 < hasamples.tst

5724-H72 (C) Copyright IBM Corp. 1994, 2024.  ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM1.

     1 : DEFINE QLOCAL(SOURCE) REPLACE
AMQ8006: IBM MQ queue created.
     2 : DEFINE QLOCAL(TARGET) REPLACE
AMQ8006: IBM MQ queue created.
     3 : DEFINE CHANNEL(CHANNEL1) CHLTYPE(SVRCONN)   TRPTYPE(TCP) MCAUSER(MUSR_MQADMIN)
REPLACE
AMQ8014: IBM MQ channel created.
     4 : DEFINE CHANNEL(CHANNEL1) CHLTYPE(CLNTCONN)  TRPTYPE(TCP) CONNAME('LOCALHOST(2345)')
QMNAME(QM1) REPLACE
AMQ8014: IBM MQ channel created.
     5 : ALTER LISTENER(SYSTEM.DEFAULT.LISTENER.TCP) TRPTYPE(TCP) PORT(2345)
AMQ8623: IBM MQ listener changed.
     6 : START LISTENER(SYSTEM.DEFAULT.LISTENER.TCP)
AMQ8021: Request to start IBM MQ Listener accepted.
     7 : START CHANNEL(CHANNEL1)
AMQ8018: Start IBM MQ channel accepted.
7 MQSC commands read.
No commands have a syntax error.
All valid MQSC commands were processed.
```

4. Zpřístupněte tabulku kanálů klienta klientům.

   Použijte datový adresář vrácený z příkazu **crtmqm** v kroku “1” na stránce 1058a přidejte do něj adresář `@ipcc` , abyste nastavili proměnnou **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.
```

amqsphac

```
Sample AMQSPHAC start
target queue is SOURCE
message Message 1
message Message 2
16:25:22 : EVENT : Connection Reconnecting (Delay: 0ms)
16:25:45 : EVENT : Connection Reconnecting (Delay: 0ms)
16:26:02 : EVENT : Connection Reconnectedmessage
Message 3
message Message 4
message Message 5
```

amqsmhac

```
Sample AMQSMHA0 start
16:25:22 : EVENT : Connection Reconnecting (Delay: 0ms)
16:25:45 : EVENT : Connection Reconnecting (Delay: 0ms)
16:26:02 : EVENT : Connection Reconnected
No more messages.
Sample AMQSMHA0 end
C:\>
```

amqsghac

```
Sample AMQSGHAC start
message Message 1
message Message 2
16:25:22 : EVENT : Connection Reconnecting (Delay: 0ms)
16:25:45 : EVENT : Connection Reconnecting (Delay: 0ms)
16:26:02 : EVENT : Connection Reconnected
message Message 3
message Message 4
message Message 5
```

Související úlohy

Ověření chování sdíleného systému souborů

Související odkazy

amqmfsck (kontrola systému souborů)

Ukázkové programy dotazování

Ukázkové programy dotazu se dotazují na některé atributy fronty pomocí volání MQINQ.

Názvy těchto programů viz “Funkce demonstrované v ukázkových programech na platformě Multiplatforms” na stránce 1025 .

Tyto programy jsou určeny ke spuštění jako spuštěné programy, takže jejich jediným vstupem je struktura MQTMC2 (zpráva spouštěče) pro IBM MQ for Multiplatforms. Tato struktura obsahuje název cílové fronty s atributy, které mají být zjišťovány. Verze C také používá název správce front. Verze COBOL používá výchozího správce front.

Aby spouštěcí proces fungoval, ujistěte se, že ukázkový program Inquire, který chcete použít, je spouštěn zprávami přicházejícími do fronty SYSTEM.SAMPLE.INQ. Chcete-li tak učinit, zadejte název ukázkového programu Inquire, který chcete použít, do pole *ApplicId* definice procesu SYSTEM.SAMPLE.INQPROCESS. IBM i Pro IBM ik tomu můžete použít příkaz CHGMQMPRC; podrobnosti viz Změna MQ Proces (CHGMQMPRC). Ukázková fronta má typ spouštěče FIRST. Pokud již existují zprávy ve frontě před spuštěním ukázky požadavku, není vzorek dotazu spuštěn odesílanými zprávami.

Po správném nastavení definice:

- ALW V případě systému AIX, Linux, and Windowsspusťte program **runmqtrm** v jedné relaci a poté spusťte program `amqsreq` v jiné relaci.
- IBM i Pro systém IBM ispusťte program AMQSERV4 v jedné relaci, pak spusťte program AMQSREQ4 v jiné relaci. Můžete použít AMQSTRG4 místo AMQSERV4, ale potenciální prodlevy při odesílání úloh by mohly způsobit, že bude méně snadné sledovat, co se děje.

Ukázkové programy požadavků použijte k odeslání zpráv požadavků, z nichž každá obsahuje pouze název fronty, do fronty SYSTEM.SAMPLE.INQ. Pro každou zprávu požadavku vzorové programy Inquire odešlou zprávu odpovědi obsahující informace o frontě uvedené ve zprávě požadavku. Odpovědi se odešlou do fronty pro odpověď uvedené ve zprávě požadavku.

IBM i V systému IBM i, pokud je ukázkový člen vstupního souboru QMQMSAMP.AMQSDATA(INQ), poslední pojmenovaná fronta neexistuje, takže ukázka vrátí zprávu sestavy s kódem příčiny selhání.

Návrh vzorového programu Inquire

Program otevře frontu uvedenou ve struktuře zprávy spouštěče, která byla předána při jejím spuštění. (Pro přehlednost toto nazýváme *frontou požadavků*.) Program používá volání MQOPEN k otevření této fronty pro sdílený vstup.

Program používá volání MQGET k odebrání 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, aby zjistil, zda se jedná o zprávu požadavku; pokud není, program zprávu vyřadí a zobrazí varovnou zprávu.

Pro každou zprávu požadavku odebranou z fronty požadavků program přečte název fronty (kterou budeme volat *cílovou frontu*). obsažené v datech a otevře tuto frontu pomocí volání MQOPEN s volbou MQOO_INQ. Program poté pomocí volání MQINQ zjišťuje hodnoty atributů *InhibitGet*, **CurrentQDepth**a **OpenInputCount** cílové fronty.

Je-li volání MQINQ úspěšné, program pomocí volání MQPUT1 vloží zprávu odpovědi do fronty pro odpověď. Tato zpráva obsahuje hodnoty tří atributů.

Pokud je volání MQOPEN nebo MQINQ neúspěšné, program použije volání MQPUT1 k vložení zprávy sestavy do fronty pro odpověď. V poli *Feedback* deskriptoru této zprávy sestavy je uveden kód příčiny vrácený voláním MQOPEN nebo MQINQ v závislosti na tom, který z nich se nezdařil.

Po volání MQINQ program zavře cílovou frontu pomocí volání MQCLOSE.

Pokud ve frontě požadavků nezbývají žádné zprávy, program tuto frontu zavře a odpojí se od správce front.

Vlastnosti dotazu ukázkového programu obslužné rutiny zpráv

AMQSIQMA je ukázkový program jazyka C pro zjišťování vlastností manipulátoru 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 poté získá jednu zprávu a dotazuje se a vytiskne všechny vlastnosti, pomocí kterých byl popisovač zprávy naplněn.

```
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 publikování/odběru demonstrují použití funkcí publikování a odběru v produktu IBM MQ.

Existují tři ukázkové programy v jazyce C, které ilustrují, jak programovat do rozhraní IBM MQ publish/subscribe. Existují některé ukázky C, které používají starší rozhraní, a existují ukázky Java . Ukázky Java používají rozhraní IBM MQ publish/subscribe v com.ibm.mq.jar a rozhraní JMS publish/subscribe v com.ibm.mqjms. Ukázky JMS nejsou v tomto tématu zahrnuty.

C

Vyhledejte ukázku vydavatele amqspub ve složce ukázek C . Spusťte jej s libovolným názvem tématu, který se vám líbí jako první parametr, následovaným volitelným názvem správce front. Například amqspub mytopic QM3 . K dispozici je také verze klienta s názvem amqspubc. Pokud se rozhodnete spustit verzi klienta, podrobnosti naleznete v části “Konfigurace správce front pro přijetí připojení klienta na platformě Multiplatforms” na stránce 1034 .

Vydavatel se připojí k výchozímu správci front a odpoví výstupem target topic is mytopic . Každý řádek, který od nynějška zadáte do tohoto okna, bude publikován v adresáři mytopic .

Otevřete jiné příkazové okno ve stejném adresáři a spusťte program odběratele amqssubse stejným názvem tématu a volitelným názvem správce front. Například amqssub mytopic QM3 .

Odběratel odpoví výstupem Calling MQGET : 30 seconds wait time . Od této chvíle se řádky, které zadáte do vydavatele, objeví ve výstupu odběratele.

Spusťte jiného odběratele v jiném příkazovém okně a sledujte, jak oba odběratelé obdrží publikování.

Úplnou dokumentaci parametrů, včetně voleb nastavení, naleznete ve zdrojovém kódu ukázky. 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 amqssbx -d mysub -t mytopic -k , chcete-li vyvolat odběratele s použitím trvalých odběrů, které jsou uchovány po ukončení odběratele.

Otestujte odběr publikováním jiné položky pomocí vydavatele. Počkejte 30 sekund na ukončení odběratele. Publikujte další položky pod stejným tématem. Restartujte odběratele. Poslední položka publikovaná v době, kdy nebyl odběratel spuštěn, je zobrazena odběratelem ihned po restartování.

Starší verze C

Existuje další sada ukázek jazyka C, které demonstrují příkazy zařazené do fronty. Některé z těchto ukázek byly původně dodány jako součást MQ0C Supportpac. Schopnosti, které ukázky demonstrují, jsou z důvodů kompatibility plně podporovány.

Odradíme vás od používání příkazového rozhraní ve frontě. Je mnohem složitější než rozhraní API pro publikování/odběr a neexistuje žádný přesvědčivý funkční důvod pro programování složitých příkazů ve frontě. Přístup zařazený do fronty však může být vhodnější, například proto, že rozhraní již používáte, nebo proto, že vaše programovací prostředí usnadňuje sestavení složité zprávy a volání generické operace MQPUT namísto vytváření různých volání MQSUB.

Další ukázky jsou umístěny v podadresáři pubsub ve složce samples .

V seznamu Tabulka 163 na stránce 1062je uvedeno šest typů ukázek.

Tabulka 163. Kategorie starších ukázkových programů publikování/odběru C

Kategorie	Programy	Komentář
RFH1	amqssr1a.c amqspr1a.c	Příklad jednoduchého publikování/odběru sestavený pomocí zpráv ve formátu RFH1 .
RFH2	amqssr2a.c amqspr2a.c	Příklad jednoduchého publikování/odběru sestavený pomocí zpráv ve formátu RFH2 .
Ukázky MQAI	amqsppca.c amqsspca.c	Jednoduchý příklad publikování/odběru sestavený pomocí příkazů PCF a rozhraní příkazů MQAI.

Tabulka 163. Kategorie starších ukázkových programů publikování/odběru C (pokračování)		
Kategorie	**Programy**	**Komentář**
MA0C Služba výsledků pomocí RFH1	`amqsgama.c` `amqsresa.c`	Služba výsledků sestavená pomocí záhlaví RFH1 1. Vyžaduje fronty definované v `amqsgama.tst` a `amqsresa.tst` . 2. `amqsresa` musí být spuštěno 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. `amqsresa` musí být spuštěno před `amqsgama`
Ukázka publikování/odběru uživatelské procedury směrování	`amqspsra.c`	Demonstruje, jak změnit cíl fronty nebo správce front pro zprávu publikování/odběru v uživatelské proceduře směrování.

Ukázkový program pro Java

Java Ukázka `MQPubSubApiSample.java` kombinuje vydavatele a odběratele v jednom programu. Jeho zdrojové a kompilované soubory tříd se nacházejí ve složce ukázek `wmqjava` .

Pokud se rozhodnete pro spuštění v režimu klienta, podrobnosti naleznete v části “Konfigurace správce front pro přijetí připojení klienta na platformě Multiplatforms” na stránce 1034 .

Spusťte ukázku z příkazového řádku pomocí příkazu Java , pokud máte nakonfigurované prostředí Java . Ukázku můžete spustit také z pracovního prostoru IBM MQ Průzkumník Eclipse , který již má nastavenou programovací pracovní plochu Java .

Možná budete muset změnit některé vlastnosti ukázkového programu, abyste jej spustili. To provedete zadáním parametrů do prostředí JVM nebo úpravou zdroje.

Pokyny v části “Spuštění ukázky MQPubSubApiSample Java” na stránce 1063 ukazují, jak spustit ukázku z pracovního prostoru Eclipse .

Spuštění ukázky MQPubSubApiSample Java
Jak spustit MQPubSubApiSample pomocí Java Vývojových nástrojů z platformy Eclipse .

Než začnete

Otevřete pracovní plochu Eclipse . Vytvořte nový adresář pracovního prostoru a vyberte jej. Zavřete uvítací okno.

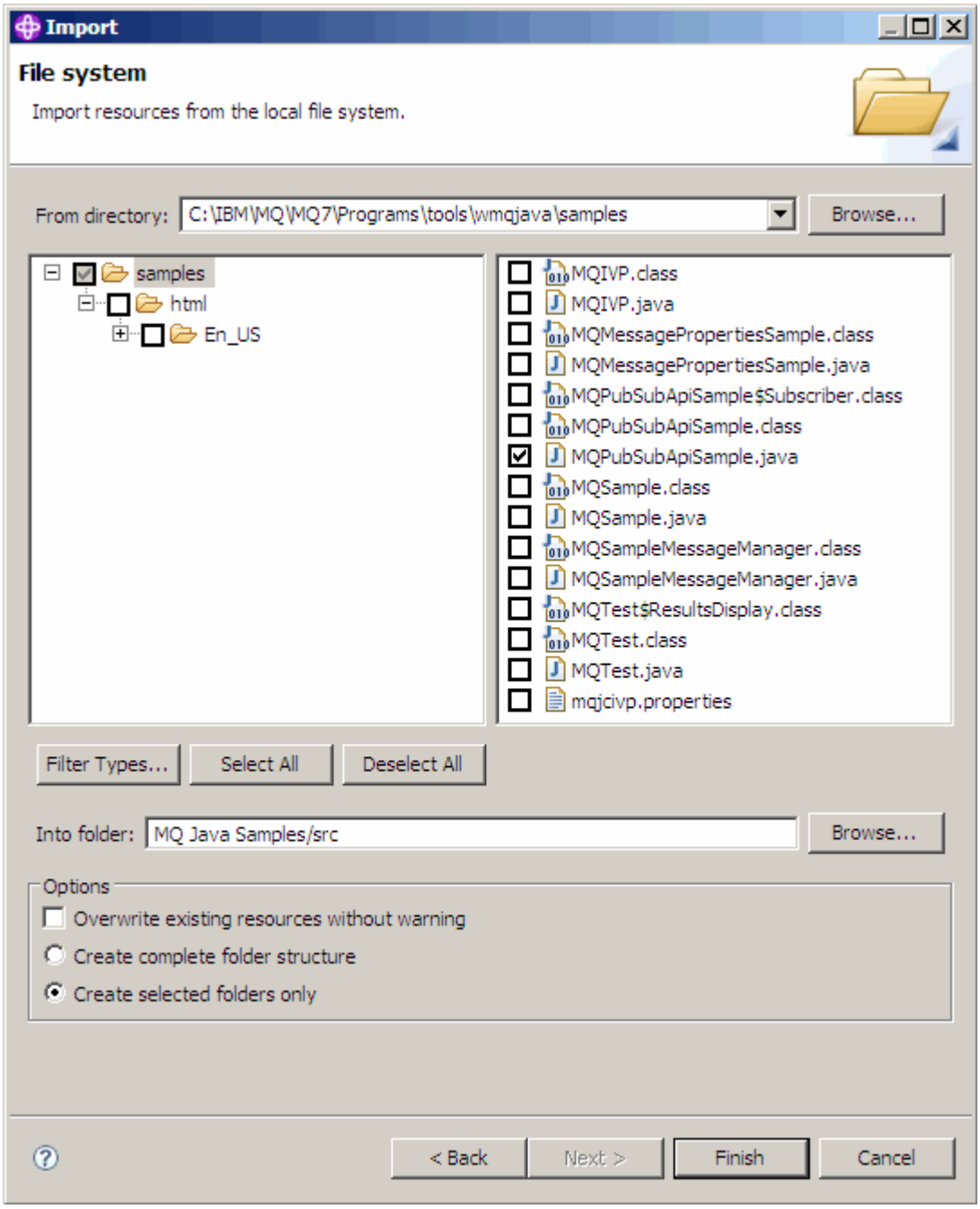
Před spuštěním jako klient postupujte podle pokynů v části “Konfigurace správce front pro přijetí připojení klienta na platformě Multiplatforms” na stránce 1034 .

Informace o této úloze

Ukázkový program Java publish/subscribe je program IBM MQ MQI client Java . Ukázka se spustí bez úprav s použitím výchozího správce front, který naslouchá na portu 1414. Tato úloha popisuje tento jednoduchý případ a obecně označuje, jak poskytnout parametry a upravit ukázku tak, aby vyhovovala různým konfiguracím produktu IBM MQ . Příklad je ilustrován spuštěním na Windows. Cesty k souborům se budou na jiných platformách lišit.

Postup

1. Import ukázkových programů Java
   a) Na pracovní ploše klepněte na volbu **Okno** > **Otevřít perspektivu** > **Ostatní** > **Java** a klepněte na tlačítko **OK**.
   b) Přepněte do pohledu **Průzkumník balíků** .
   c) Klepněte pravým tlačítkem myši na mezeru v pohledu **Průzkumník balíků** . Klepněte na volbu **Nový** > **Java projekt**.
   d) Do pole **Project name** zadejte typ `MQ Java Samples`. Klepněte na tlačítko **Další**.
   e) Na panelu **Java Settings** přepněte na kartu **Knihovny** .
   f) Klepněte na volbu **Přidat externí soubory JAR**.
   g) Přejděte na *MQ_INSTALLATION_PATH* `\java\lib` , kde *MQ_INSTALLATION_PATH* je instalační složka IBM MQ a vyberte `com.ibm.mq.jar` a `com.ibm.mq.jmqi.jar`
   h) Klepněte na tlačítko **Otevřít** > **Dokončit.**
   i) Klepněte pravým tlačítkem myši na `src` v pohledu **Průzkumník balíků** .
   j) Vyberte volbu **Importovat ...** > **Obecné** > **Systém souborů** > **Další** > **Procházet**... a přejděte na cestu *MQ_INSTALLATION_PATH* `\tools\wmqjava\samples` , kde *MQ_INSTALLATION_PATH* je instalační adresář IBM MQ .
   k) Na panelu **Importovat** Obrázek 130 na stránce 1065klepněte na volbu `samples` (nezaškrtávejte políčko).
   l) Vyberte volbu `MQPubSubApiSample.java`. Pole **Into folder** by mělo obsahovat řetězec `MQ Java Samples/src`. Klepněte na volbu **Dokončit**.



Obrázek 130. Import systému souborů

2. Spusťte ukázkový program publikování/odběru.
   Existují dva způsoby spuštění programu v závislosti na tom, zda potřebujete změnit výchozí parametry.
   - První volba spustí program bez provedení změn:
     - V hlavní nabídce pracovního prostoru rozbalte složku `src` . Klepněte pravým tlačítkem myši na položku **MQPubSubApiSample.java Spustit jako** > **1. Java Aplikace**
   - Druhá volba spustí program s parametry nebo s upraveným zdrojovým kódem pro vaše prostředí:
     - Otevřete soubor `MQPubSubApiSample.java` a studujte konstruktor `MQPubSubApiSample` .
     - Upravte atributy programu.

Tyto atributy lze upravit pomocí přepínače -D JVM nebo poskytnutím výchozí hodnoty pro System property úpravou zdrojového kódu.

- topicObject
- QueueManagerName
- subscriberCount

Tyto atributy lze měnit pouze úpravou zdrojového kódu v konstruktoru.

- hostname
- Port
- kanál

Chcete-li nastavit System properties, kódujte výchozí hodnotu v přistupujícím objektu, například:

```
queueManagerName = System.getProperty("com.ibm.mq.pubSubSample.queueManagerName",
"QM3");
```

Nebo zadejte parametr do prostředí JVM pomocí volby `-D` , jak ukazuje následující postup:

a. Zkopírujte úplný název 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 části **Vytvořit, spravovat a spustit aplikace** a klepněte na kartu **(x) = Argumenty** .
c. V podokně **Argumenty VM:** zadejte `-D` a vložte název System.property , `com.ibm.mq.pubSubSample.queueManagerName`, následovaný `=QM3`. Klepněte na tlačítko **Použít** > **Spustit**.
d. Přidejte další argumenty jako seznam oddělený čárkami nebo jako další řádky v podokně bez oddělovačů čárky.

   Například: `-Dcom.ibm.mq.pubSubSample.queueManagerName=QM3`, `-Dcom.ibm.mq.pubSubSample.subscriberCount=6`.

Ukázkový program uživatelské procedury publikování

AMQSPSE0 je ukázkový program jazyka C uživatelské procedury pro zachycení publikování před jeho doručením odběrateli. Uživatelská procedura pak může například změnit záhlaví zprávy, informační obsah nebo místo určení nebo zabránit publikování zprávy na odběratele.

Chcete-li spustit ukázku, proveďte následující úlohy:

1. Konfigurujte správce front:
   - Linux AIX Na systémech AIX and Linux přidejte sekci podobnou této do souboru `qm.ini` :

     ```
     PublishSubscribe:
     PublishExitPath=Module
     PublishExitFunction=EntryPoint
     ```

     kde modul je *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 IBM MQ.
3. Restartujte správce front, aby se konfigurace vybrala.
4. V procesu aplikace, který má být trasován, popište, kam se mají zapisovat trasovací soubory. Příklad:

- Linux AIX Na systémech AIX and Linux se ujistěte, že adresář `/var/mqm/trace` existuje, a exportujte proměnnou prostředí **MQPSE_TRACE_LOGFILE** :

```
export MQPSE_TRACE_LOGFILE=/var/mqm/trace/PubTrace
```

- Windows V systému Windowsse ujistěte, že adresář `C:\temp` existuje, a nastavte proměnnou prostředí **MQPSE_TRACE_LOGFILE** :

```
set MQPSE_TRACE_LOGFILE=C:\temp\PubTrace
```

Ukázkové programy Put

Ukázkové programy Put vložily zprávy do fronty pomocí volání MQPUT.

Názvy těchto programů viz “Funkce demonstrované v ukázkových programech na platformě Multiplatforms” na stránce 1025 .

Návrh vzorového programu Put

Program používá volání MQOPEN s volbou MQOO_OUTPUT k otevření cílové fronty pro vkládání zpráv.

Pokud nemůže otevřít frontu, program zobrazí chybovou zprávu obsahující kód příčiny vrácený voláním MQOPEN. Chcete-li zachovat jednoduchost programu, při tomto a následných voláních MQI použije program pro mnoho voleb výchozí hodnoty.

Pro každý řádek vstupu program načte text do vyrovnávací paměti a použije volání MQPUT k vytvoření zprávy datagramu obsahující text tohoto řádku. Program pokračuje, dokud nedosáhne konce vstupu nebo dokud neselže volání MQPUT. Pokud program dosáhne konce vstupu, uzavře frontu pomocí volání MQCLOSE.

Spuštění ukázkových programů Put

Spuštění ukázek amqsput a amqsputc

ALW

Ukázka **amqqsput** je program pro vkládání zpráv pomocí lokálních vazeb a ukázka **amqsputc** je program pro vkládání zpráv pomocí vazeb klienta. Každý z těchto programů má následující poziční parametry:

1. Název cílové fronty (povinné)
2. Název správce front (volitelné)

   Není-li určen správce front, produkt **amqsput** se připojí k výchozímu správci front a produkt **amqsputc** se připojí ke správci front určenému proměnnou prostředí *MQSERVER* nebo souborem definice kanálu klienta.
3. Volby otevření (volitelné)

   Nejsou-li zadány volby otevření, ukázka použije hodnotu 8208, což je kombinace těchto dvou voleb:

   - MQOO_OUTPUT
   - MQOO_FAIL_IF_QUIESCING
4. Volby zavření (volitelné)

   Nejsou-li zadány volby zavření, ukázka použije hodnotu 0, která je MQCO_NONE.
5. Název cílového správce front (volitelné)

   Pokud není určen cílový správce front, pole `ObjectQMgrName` v produktu MQOD bude ponecháno prázdné.
6. Název dynamické fronty (volitelné)

Není-li uveden název dynamické fronty, pole `DynamicQName` v MQOD bude ponecháno prázdné.

Pomocí následujících proměnných prostředí zadejte pověření, která se používají k ověření se správcem front:

MQSAMP_USER_ID
Nastavte na ID uživatele, které se má použít pro ověření připojení, pokud chcete použít ID uživatele a heslo pro ověření u správce front. Program vyzve k zadání hesla, které má doprovázet ID uživatele.

Linux V 9.3.4 AIX **MQSAMP_TOKEN**
Nastavte na neprázdnou hodnotu, chcete-li zadat token ověření pro ověření u správce front. Program vyzve k zadání tokenu ověření. Tokeny ověřování mohou být použity pouze ukázkou **amqsputc** , která používá vazby klienta.

Chcete-li spustit tyto programy, zadejte jeden z následujících příkazů:

- `amqsput` *myqueue qmanagername*
- `amqsputc` *myqueue qmanagername*

kde *myqueue* je název fronty, do které budou zprávy vloženy, a *qmanagername* je správce front, který vlastní *myqueue*.

Spuštění ukázky amq0put

ALW

Verze COBOL nemá žádné parametry. Připojí se k výchozímu správci front a po jeho spuštění 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 neexistují žádná další data.

Spuštění ukázky C AMQSPUT4 (IBM i)

IBM i

Program v jazyce C AMQSPUT4, který je k dispozici pouze pro platformu IBM i , vytváří zprávy čtením dat ze členu zdrojového souboru.

Při spuštění programu musíte uvést 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 vkládané ukázky je dodávána v knihovně QMQMSAMP soubor AMQSDATA člen PUT.

Poznámka: Pamatujte, že názvy front rozlišují velká a malá písmena. Všechny fronty vytvořené ukázkovým programem pro vytvoření souboru AMQSAMP4 mají názvy vytvořené velkými písmeny.

Program v jazyce C vkládá 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 zpráv datagramu a zastaví se, když přečte prázdný řádek na konci souboru.

Pomocí ukázkového datového souboru je příkaz:

```
CALL PGM(QMQM/AMQSPUT4) PARM('QMQMSAMP/AMQSDATA(PUT)')
```

Spuštění ukázky AMQ0PUT4 COBOL (IBM i)

IBM i

Program v jazyce COBOL AMQ0PUT4, dostupný pouze na platformě IBM i , vytváří zprávy přijímáním dat z klávesnice.

Chcete-li spustit program, zavolejte program a jako parametr programu zadejte název cílové fronty. Program přijme vstup z klávesnice do bufferu a vytvoří zprávu datagramu pro každý řádek textu. Program se zastaví po zadání prázdného řádku na klávesnici.

Ukázkové programy referenčních zpráv

Ukázky referenční zprávy umožňují přenos velkého objektu z jednoho uzlu do jiného (obvykle na různých systémech), aniž by byl objekt uložen ve frontách systému IBM MQ na zdrojovém nebo cílovém uzlu.

K dispozici je sada ukázkových programů, která demonstruje, jak mohou být referenční zprávy vloženy do fronty, přijaty pomocí uživatelských procedur pro zprávy a převzaty z fronty. Ukázkové programy používají k přesouvání souborů referenční zprávy. Chcete-li přesunout jiné objekty, jako jsou databáze, nebo chcete-li provést kontroly zabezpečení, definujte vlastní uživatelskou proceduru na základě ukázky amqsxrm.

Verze ukázkového programu uživatelské procedury pro referenční zprávu, která se má použít, závisí na platformě, na které je kanál spuštěn:

- Na všech platformách použijte příkaz amqsxrma na odesílajícím konci.
- Použijte amqsxrma na přijímacím konci, pokud je přijímač spuštěn na libovolné platformě kromě platformy IBM i.
- IBM i Pokud je zásobník spuštěn v adresáři IBM i, použijte příkaz amqsxrm4.

.

IBM i *Poznámky pro uživatele IBM i*

Chcete-li přijmout referenční zprávu pomocí vzorové uživatelské procedury pro zprávy, uveďte soubor v kořenovém systému souborů IFS nebo v jakémkoli podadresáři, aby bylo možné vytvořit proudový soubor.

Ukázková uživatelská procedura pro zprávy v systému IBM i vytvoří soubor, převede data na EBCDIC a nastaví kódovou stránku na vaši systémovou kódovou stránku. Pak můžete tento soubor zkopírovat do QSYS.LIB používající příkaz CPYFRMSTMF. Příklad:

```
CPYFRMSTMF FROMSTMF('JANEP/TEST.TXT')
TOMBR('qsys.lib.janep.lib/test.fie/test.mbr') MBROPT(*REPLACE)
CVTDTA(*NONE)
```

Příkaz CPYFRMSTMF nevytváří soubor. Před spuštěním tohoto příkazu jej musíte vytvořit.

Pokud odešlete soubor z QSYS.LIB, v ukázkách se nepožadují žádné změny. U všech ostatních systémů souborů se ujistěte, že CCSID uvedený v poli CodedCharSetId ve struktuře MQRMH odpovídá datům, která odesíláte.

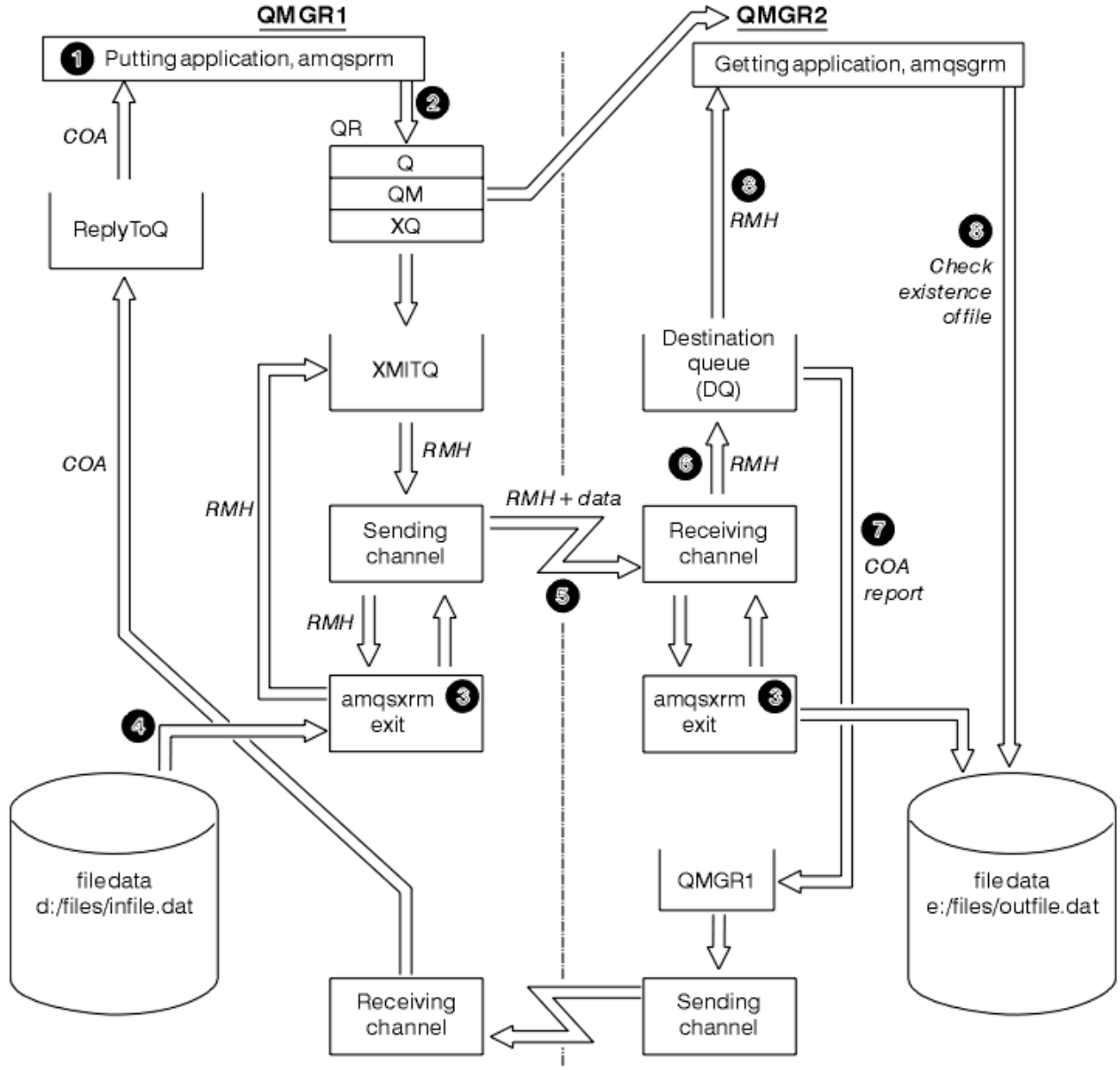
Když používáte integrovaný systém souborů, vytvořte programové moduly s nastavenou volbou SYSIFCOPT (*IFSIO). Chcete-li přesunout databázové soubory nebo soubory záznamů s pevnou délkou, definujte vlastní uživatelskou proceduru na základě dodané ukázky AMQSXRM4.

Doporučenou metodou přenosu databázového souboru je převést jej na strukturu IFS pomocí příkazu CPYTOSTMF a poté odeslat referenční zprávu připojující soubor IFS. Pokud se rozhodnete přenést databázový soubor odkazem na něj z IFS, ale nepřevedete jej na strukturu IFS, musíte uvést název členu. Pokud zvolíte tuto metodu, není zaručena integrita dat.

Multi *Spuštění ukázek referenční zprávy*

Pomocí tohoto příkladu zjistíte, jak spustit ukázkovou aplikaci Referenční zpráva AMQSPRM na systému AIX, Linux, and Windowsnebo AMQSPRMA na systému IBM i. Příklad ukazuje, jak mohou být referenční zprávy vloženy do fronty, přijaty pomocí uživatelských procedur pro zprávy a převzaty z fronty.

Ukázky referenční zprávy jsou spuštěny takto:



Obrázek 131. Spuštění ukázek referenční zprávy

1. Nastavte prostředí pro spouštění modulů listener, kanálů a monitorů spouštěčů a definujte kanály a fronty.

   Pro účely popisu způsobu nastavení referenční zprávy se tento příklad odkazuje na odesílající počítač jako na MACHINE1 se správcem front s názvem QMGR1 a přijímající počítač jako MACHINE2 se správcem front s názvem QMGR2.

   Poznámka: Následující definice umožňují sestavení referenční zprávy pro odeslání souboru s typem objektu FLATFILE ze správce front QMGR1 do QMGR2 a opětovné vytvoření souboru, jak je definováno ve volání AMQSPRM (nebo AMQSPRMA na IBM i). Referenční zpráva (včetně dat souboru) se odešle pomocí kanálu CHL1 a přenosové fronty XMITQ a umístí se do fronty DQ. Sestavy výjimek a COA jsou odeslány zpět do QMGR1 pomocí kanálu REPORT a přenosové fronty QMGR1.

Aplikace, která přijme referenční zprávu (AMQSGRM nebo AMQSGRMA on IBM i), se spustí pomocí 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 .

IBM i Definice MQSC používaly pro definování uživatelských procedur styl AIX , takže pokud používáte MQSC na systému IBM i, musíte je odpovídajícím způsobem upravit. Je důležité si uvědomit, že data zprávy FLATFILE rozlišují velká a malá písmena a ukázka nebude fungovat, pokud nebude napsána 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 **IBM i syntaxe příkazu**

Poznámka: Pokud nezadáte 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)
```

IBM i **IBM i syntaxe příkazu**

Poznámka: IBM i V systému IBM iplatí, že pokud nezadáte 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 IBM MQ :
   a. Kde je to možné pro platformu, spusťte modul listener pro odesílající a přijímající správce front.
   b. Spuštění kanálů CHL1 a REPORT
   c. V přijímajícím správci front spusťte monitor spouštěčů pro inicializační frontu INITQ.
3. Vyvolejte ukázkový program AMQSPRM (AMQSPRMA na IBM i) z příkazového řádku pomocí následujících parametrů:

   -m
   Název lokálního správce front; výchozí nastavení 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 fronta definovaná v parametru -q existuje. Standardně se jedná o správce front určeného v parametru -m.

   -t
   Typ objektu

   -w
   Interval čekání, tj. doba čekání na výjimku a sestavy COA od přijímajícího správce front

   Chcete-li například použít ukázku s objekty definovanými dříve, použijte následující parametry:

```
-mQMGR1 -iInput File -oOutput File -qQR -tFLATFILE -w120
```

   Zvýšení čekací doby umožňuje, aby byl velký soubor odeslán po síti dříve, než dojde k vypršení časového limitu pro vložení zpráv.

```
amqsprm -q QR -m QMGR1 -i d:\x\file.in -o d:\y\file.out -t FLATFILE
```

   Uživatelé systému IBM i: IBM i V systému IBM ipostupujte takto:

   a. Zadejte následující příkaz:

```
CALL    PGM(QMQM/AMQSPRM4) PARM('-mQMGR1' +
'-i/refmsgs/rmsg1' +
'-o/refmsgs/rmsgx' '-qQR' +
'-gQMGR1' '-tFLATFILE' '-w15')
```

To předpokládá, 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, nikoli pomocí kořenového adresáře.

c. Když voláte 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 V systému IBM imůžete při zadávání parametrů použít dopředné lomítko (/) nebo pomlčku (-). Příklad:

```
amqsprm /i d:\files\infile.dat /o e:\files\outfile.dat /q QR
/m QMGR1 /w 30 /t FLATFILE
```

Linux AIX Na platformách AIX and Linux musíte místo jednoho použít dvě zpětná lomítka (\\), která označují 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 put Reference Message 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 vzorový příkaz.
- Pokud je fronta QR vzdálenou frontou, referenční zpráva je odeslána jinému správci front na jiném systému, kde je vytvořen soubor s názvem a cestou `e:\files\outfile.dat`. Obsah tohoto souboru je stejný jako obsah zdrojového souboru.
- amqsprm čeká 30 sekund na sestavu COA z cílového správce front.
- Typ objektu je `flatfile`, takže kanál použitý k přesunu zpráv z fronty QR musí tuto hodnotu uvést v poli *MsgData* .

4. Když definujete své kanály, vyberte uživatelskou proceduru zprávy na obou odesílajících a přijímacích koncích, aby byla amqsxrm.

Windows To je definováno v systému Windows takto:

```
msgexit(' pathname\amqsxrm.dll(MsgExit)')
```

Linux AIX To je definováno v systému AIX and Linux takto:

```
msgexit(' pathname/amqsxrm(MsgExit)')
```

Zadáte-li název cesty, zadejte úplný název. Pokud vynecháte název cesty, předpokládá se, že program je v cestě uvedené v souboru `qm.ini` (nebo v systému IBM MQ for Windowsv cestě uvedené v registru).

5. Uživatelská procedura kanálu přečte záhlaví referenční zprávy a nalezne soubor, na který odkazuje.
6. Uživatelská procedura kanálu pak může soubor segmentovat před jeho odesláním po kanálu spolu s hlavičkou.

Linux AIX V systému AIX and Linuxzměňte vlastníka skupiny cílového adresáře na 'mqm', aby ukázková uživatelská procedura pro zprávy mohla vytvořit soubor v tomto adresáři. Změňte také oprávnění cílového adresáře tak, aby do něj členové skupiny mqm mohli zapisovat. Data souboru nejsou uložena ve frontách IBM MQ .

7. Je-li poslední segment souboru zpracován uživatelskou procedurou pro příjem zpráv, je referenční zpráva vložena do cílové fronty určené parametrem amqsprm. Je-li tato fronta spuštěna (tj. definice uvádí atributy fronty **Trigger**, **InitQ**a **Process**), spustí se program uvedený parametrem PROC cílové fronty. Program, který se má spustit, musí být definován v poli ApplId atributu **Process** .
8. Když referenční zpráva dosáhne cílové fronty (DQ), odešle se zpět do vkládající aplikace (amqsprm) sestava COA.
9. Ukázka Získat referenční zprávu, amqsgrm, získá zprávy z fronty určené ve vstupní spouštěcí zprávě a zkontroluje existenci souboru.

Návrh ukázky Vložit referenční zprávu (amqsprma.c, AMQSPRM4)
V tomto tématu je uveden podrobný popis ukázky Vložit referenční zprávu.

Tato ukázka vytvoří referenční zprávu, která odkazuje na soubor a umístí jej do určené fronty:

1. Ukázka se připojí k lokálnímu správci front pomocí MQCONN.
2. Poté otevře (MQOPEN) modelovou frontu, která se používá pro příjem zpráv sestavy.
3. Ukázka sestaví referenční zprávu obsahující hodnoty nezbytné pro přesun souboru, například názvy zdrojových a cílových souborů a typ objektu. Příklad: Ukázka dodávaná s produktem IBM MQ sestaví referenční zprávu pro odeslání souboru `d:\x\file.in` z `QMGR1` do `QMGR2` a pro opětovné vytvoření souboru jako `d:\y\file.out` pomocí 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 AIX and Linux použijte dvě zpětná lomítka (\\) namísto jednoho k označení adresáře cílového souboru. 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 dat souboru) do fronty určené parametrem /q. Pokud se jedná o vzdálenou frontu, zpráva se vloží do odpovídající přenosové fronty.
5. Ukázka čeká po dobu určenou v parametru /w (výchozí hodnota je 15 sekund) na sestavy COA, které jsou spolu se sestavami výjimek odeslány zpět do dynamické fronty vytvořené v lokálním správci front (QMGR1).

Návrh ukázky Konec referenční zprávy (amqsxrma.c, AMQSXRM4)
Tato ukázka rozpoznává referenční zprávy s typem objektu, který odpovídá typu objektu v poli uživatelských dat uživatelské procedury pro zprávu v definici kanálu.

Pro tyto zprávy se stane následující:

- V kanálu odesilatele nebo serveru se zadaná délka dat zkopíruje ze zadaného posunutí uvedeného souboru do prostoru zbývajícího ve vyrovnávací paměti agenta po referenční zprávě. Pokud není dosaženo konce souboru, referenční zpráva se po aktualizaci pole *DataLogicalOffset* vloží zpět do přenosové fronty.
- Pokud je u žadatele nebo přijímacího kanálu pole *DataLogicalOffset* nula a uvedený soubor neexistuje, vytvoří se. Data následující za referenční zprávou se přidají na konec uvedeného souboru. Pokud referenční zpráva není poslední pro uvedený soubor, je vyřazena. Jinak je vrácena uživatelské proceduře kanálu bez připojených dat, která mají být vložena do cílové fronty.

Pokud je pro kanály odesilatele a serveru pole *DataLogicalLength* ve vstupní referenční zprávě nula, zbývající část souboru od *DataLogicalOffset* do konce souboru se odešle podél kanálu. Pokud není nula, odešle se pouze uvedená délka.

Pokud dojde k chybě (například pokud ukázku nelze otevřít soubor), MQCXP. Parametr *ExitResponse* je nastaven na hodnotu MQXCC_SUPPRESS_FUNCTION, takže zpracovávaná zpráva je vložena do fronty nedoručených zpráv namísto pokračování do cílové fronty. V prostředí MQCXP je vrácen kód zpětné vazby. *Feedback* a vrátil se do aplikace, která vložila zprávu do pole *Feedback* deskriptoru

zprávy sestavy. Důvodem je, že vkládající aplikace požadovala zprávy o výjimce nastavením parametru MQRO_EXCEPTION v poli *Report* deskriptoru MQMD.

Pokud se kódování nebo *CodedCharacterSetId* (CCSID) referenční zprávy liší od kódování správce front, referenční zpráva se převede na lokální kódování a CCSID. V našem vzorku, amqsprm, je formát objektu MQFMT_STRING, takže amqsxrm převádí data objektu na lokální CCSID na přijímacím konci před tím, než jsou data zapsána do souboru.

Neuvádějte formát přenášeného souboru jako MQFMT_STRING, pokud soubor obsahuje vícebajtové znaky (například DBCS nebo Unicode). Je to proto, že vícebajtový znak může být rozdělen, když je soubor segmentován na odesílajícím konci. Chcete-li přenést a převést takový soubor, uveďte jiný formát než MQFMT_STRING, aby jej uživatelská procedura referenční zprávy nepřevedl a nepřevedl soubor na přijímající straně po dokončení přenosu.

Kompilace ukázky uživatelské procedury referenční zprávy
Chcete-li kompilovat ukázku uživatelské procedury 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
```

zapIBM i

IBM i

```
CRTCMOD MODULE(MYLIB/AMQSXRMA) SRCFILE(QMQMSAMP/QCSRC)
TERASPACE(*YES *TSIFC)
```

Poznámka:

1. Chcete-li vytvořit 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
```

zapWindows

Windows Produkt IBM MQ nyní dodává knihovně mqm balíky klienta i balíky serveru, takže následující příklad používá `mqm.lib` místo `mqmvx.lib`:

```
cl amqsqrma.c /link /out:amqsxrm.dll /dll mqm.lib mqm.lib /def:amqsxrm.def
```

Související pojmy

“Psaní programů uživatelské procedury kanálu” na stránce 931
Následující informace vám mohou pomoci při psaní programů pro uživatelské procedury kanálu.

Návrh ukázky Získat referenční zprávu (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 určenému správci front pomocí MQCONN a otevře určenou frontu pomocí MQOPEN.
3. Ukázka vydá příkaz MQGET s intervalem čekání 15 sekund v rámci smyčky pro získání zpráv z fronty.
4. Je-li zpráva referenční zprávou, ukázka zkontroluje existenci přeneseného souboru.
5. Poté uzavře frontu a odpojí se od správce front.

Ukázkové programy požadavků

Ukázkové programy požadavků demonstrují zpracování typu klient/server. Ukázky jsou klienti, kteří vkládají zprávy požadavků do fronty cílového serveru, která je zpracována programem serveru. Čekají, až program serveru vloží zprávu odpovědi do fronty pro odpověď.

Ukázky požadavků vložily řadu zpráv požadavků do fronty cílového serveru pomocí volání MQPUT. Tyto zprávy určují lokální frontu SYSTEM.SAMPLE.REPLY jako frontu pro odpověď, což může být lokální nebo vzdálená fronta. Programy čekají na zprávy s odpovědí a pak je zobrazí. Odpovědi jsou odesílány pouze v případě, že je fronta cílového serveru zpracovávána serverovou aplikací, nebo pokud je pro tento účel spuštěna aplikace (ukázkové programy Inquire, Set a Echo jsou navrženy tak, aby byly spuštěny). Ukázka C čeká 1 minutu (ukázka COBOL čeká 5 minut), na doručení první odpovědi (aby byl čas na spuštění serverové aplikace) a 15 sekund na následné odpovědi, ale obě ukázky mohou skončit bez získání odpovědí. Názvy ukázkových programů požadavků viz “Funkce demonstrované v ukázkových programech na platformě Multiplatforms” na stránce 1025 .

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 (volitelné)
3. Fronta odpovědí (volitelné)

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 název fronty odpovědí.

Pokud vynecháte název správce front, předpokládá se, že frontu vlastní výchozí správce front. Pokud vynecháte název fronty odpovědí, bude poskytnuta výchozí fronta odpovědí.

Spuštění ukázky amq0req0.cbl

Verze COBOL nemá žádné parametry. Připojí se k výchozímu správci front a po jeho spuštění budete vyzváni:

```
Please enter the name of the target server queue
```

Program vezme svůj vstup z StdIn a přidá každý řádek do fronty cílového serveru, přičemž každý řádek textu vezme jako obsah zprávy požadavku. Program skončí, když se přečte řádek s hodnotou null.

Spuštění ukázky AMQSREQ4

Program v jazyce C vytváří zprávy převzetím dat ze stdin (klávesnice) s prázdným časem ukončení vstupu. Program má až tři parametry: název cílové fronty (povinné), název správce front (volitelné) a název fronty pro odpověď (volitelné). Není-li zadán žádný název správce front, použije se výchozí správce front. Není-li uvedena žádná fronta pro odpověď, SYSTEM.SAMPLE.REPLY .

Zde je uveden příklad volání ukázkového programu jazyka C s určením fronty pro odpověď, ale ponechání výchozího nastavení správce front:

```
CALL PGM(QMQM/AMQSREQ4) PARM('SYSTEM.SAMPLE.LOCAL' '' 'SYSTEM.SAMPLE.REPLY')
```

Poznámka: Pamatujte, že názvy front rozlišují velká a malá písmena. Všechny fronty vytvořené ukázkovým programem pro vytvoření souboru AMQSAMP4 mají názvy vytvořené velkými písmeny.

Spuštění ukázky AMQ0REQ4

Program v jazyce COBOL vytváří zprávy přijímáním dat z klávesnice. Chcete-li spustit program, zavolejte program a uveďte název cílové fronty jako parametr. Program přijme vstup z klávesnice do bufferu a vytvoří zprávu požadavku pro každý řádek textu. Program se zastaví po zadání prázdného řádku na klávesnici.

Spuštění ukázky požadavku pomocí spouštěče

Pokud je ukázka použita se spouštěcím programem a jedním z ukázkových programů Inquire, Set nebo Echo, řádek vstupu musí být název fronty, ke které má spouštěný program přistupovat.

ALW *Spuštění ukázky požadavku pomocí spouštěče na AIX, Linux, and Windows*
V systému AIX, Linux, and Windowsspusťte program monitoru spouštěčů RUNMQTRM v jedné relaci a poté spusťte program amqsreq v jiné relaci.

Chcete-li spustit ukázky pomocí spouštěče, postupujte takto:

1. Spusťte program monitoru spouštěčů RUNMQTRM v jedné relaci (inicializační fronta SYSTEM.SAMPLE.TRIGGER je k dispozici pro použití).
2. Spusťte program amqsreq v jiné relaci.
3. Ujistěte se, že jste definovali frontu cílového serveru.

   Ukázkové fronty, které máte k dispozici pro použití jako fronta cílového serveru pro ukázku požadavku na vložení zpráv, jsou:

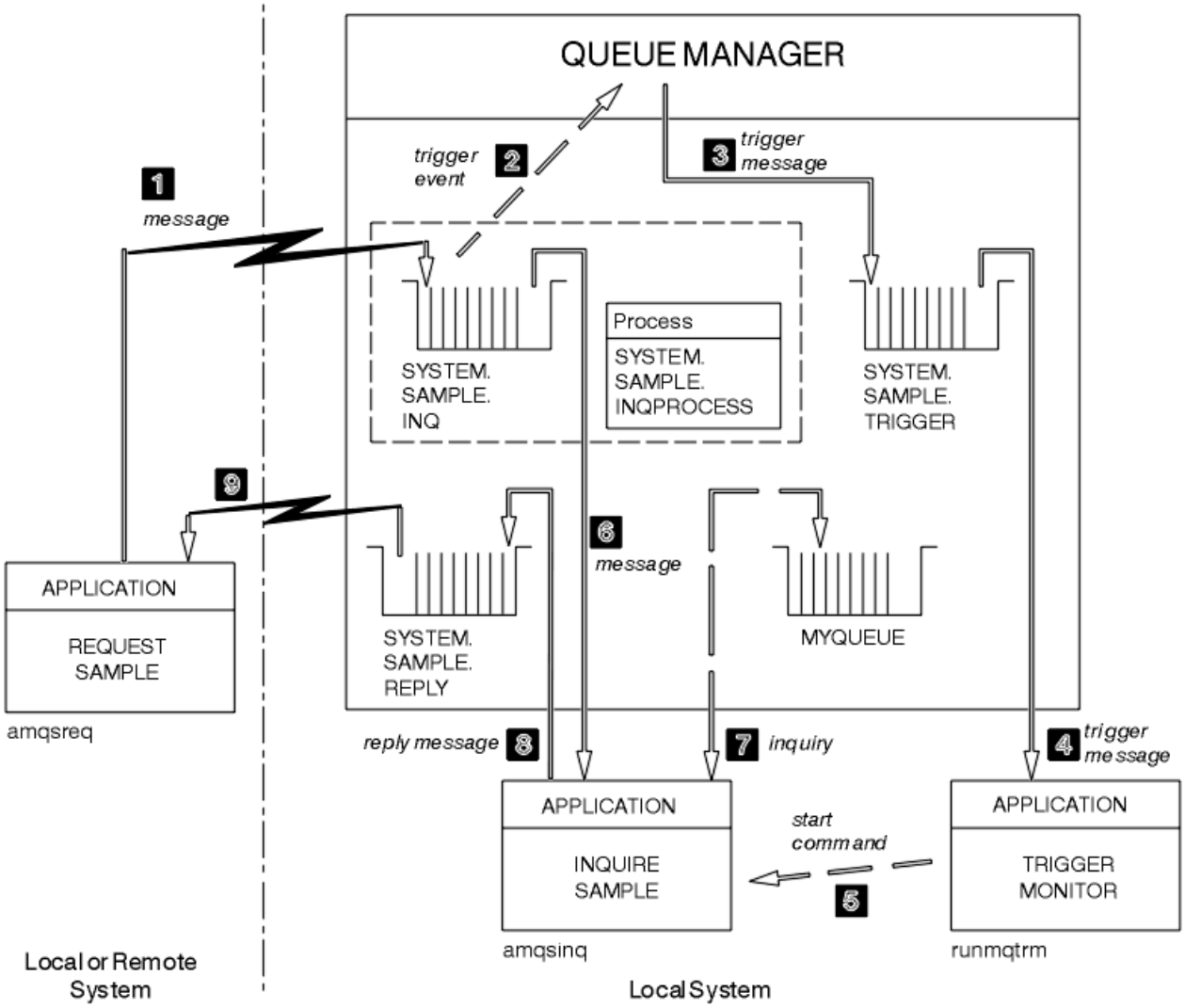
   - SYSTEM.SAMPLE.INQ -pro ukázkový program Inquire
   - SYSTEM.SAMPLE.SET -pro ukázkový program Set
   - SYSTEM.SAMPLE.ECHO -pro ukázkový program Echo

   Tyto fronty mají typ spouštěče FIRST, takže pokud již existují zprávy ve frontách před spuštěním ukázky Požadavek, nejsou serverové aplikace spouštěny odesílanými zprávami.
4. Ujistěte se, že jste definovali frontu pro ukázkový program Inquire, Set nebo Echo, který se má použít.

To znamená, že monitor spouštěčů je připraven, když ukázka požadavku odešle zprávu.

Poznámka: Ukázkové definice procesů vytvořené pomocí příkazu RUNMQSC a souboru amqscos0.tst spouštějí ukázky jazyka C. Změňte definice procesů v souboru amqscos0.tst a použijte příkaz RUNMQSC s tímto aktualizovaným souborem pro použití verzí jazyka COBOL.

Produkt Obrázek 132 na stránce 1078 demonstruje, jak společně používat ukázky požadavku a dotazování.



Obrázek 132. Požadovat a dotazovat se na vzorky pomocí spouštěče

V souboru Obrázek 132 na stránce 1078 ukázka požadavku vkládá zprávy do fronty cílového serveru, SYSTEM.SAMPLE.INQa vzorek Inquire dotazují frontu MYQUEUE. Případně můžete použít jednu z ukázkových front definovaných při spuštění příkazu amqscos0.tstnebo libovolnou jinou frontu, kterou jste definovali, pro ukázku dotazování.

Poznámka: Čísla v souboru Obrázek 132 na stránce 1078 zobrazují posloupnost událostí.

Chcete-li spustit ukázky požadavku a dotazu pomocí spouštěče:

1. Zkontrolujte, zda jsou definovány fronty, které chcete použít. Spuštěním příkazu amqscos0.tstdefinujte 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 spuštěny na stejné platformě, musí všechny procesy spuštěné monitorem spouštěčů definovat *ApplType*, jinak server

použije 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 má ukázka dotazování používat:

```
MYQUEUE
```

5. Zadejte prázdný řádek (pro ukončení programu požadavku).
6. Ukázka požadavku pak zobrazí zprávu obsahující data, která program Inquire získal z MYQUEUE.

Můžete použít více než jednu frontu; v tomto případě zadejte názvy ostatních front v kroku “4” na stránce 1079.

Další informace o spouštění viz “Spuštění aplikací IBM MQ pomocí spouštěčů” na stránce 836.

IBM i *Spuštění ukázky požadavku pomocí spouštěče na IBM i*

V systému IBM ispusťte ukázkový spouštěcí server AMQSERV4v jedné úloze, pak spusťte AMQSREQ4 v jiné úloze. To znamená, že spouštěcí server je připraven, když ukázkový program požadavku odešle zprávu.

Poznámka:

1. Ukázkové definice vytvořené pomocí AMQSAMP4 spustí verze C ukázek. Chcete-li spustit verze jazyka COBOL, změňte definice procesu SYSTEM.SAMPLE.ECHOPROCESS, SYSTEM.SAMPLE.INQPROCESSa SYSTEM.SAMPLE.SETPROCESS. Můžete použít příkaz CHGMQMPRC (podrobnosti viz Change MQ Process (CHGMQMPRC)). , chcete-li tak učinit, nebo upravte a spusťte vlastní verzi produktu AMQSAMP4.
2. Zdrojový kód pro AMQSERV4 je dodáván pouze pro jazyk C. Kompilovaná verze (kterou můžete použít s ukázkami v jazyce COBOL) je však dodávána v knihovně QMQM.

Zprávy požadavků můžete vložit do těchto ukázkových front serveru:

- SYSTEM.SAMPLE.ECHO (pro ukázkové programy Echo)
- SYSTEM.SAMPLE.INQ (pro ukázkové programy Inquire)
- SYSTEM.SAMPLE.SET (pro ukázkové programy Set)

Vývojový diagram pro SYSTEM.SAMPLE.ECHO je zobrazen v souboru Obrázek 133 na stránce 1081. Pomocí ukázkového datového souboru je příkaz k vydání požadavku programu C na tento server:

```
CALL PGM(QMQMSAMP/AMQSREQ4) PARM('QMQMSAMP/AMQSDATA(ECHO)')
```

Poznámka: Tato vzorová fronta má typ spouštěče FIRST, takže pokud již existují zprávy ve frontě před spuštěním ukázky Požadavek, nejsou serverové aplikace spouštěny odesílanými zprávami.

Chcete-li se pokusit o další příklady, můžete zkusit následující variace:

- Použijte AMQSTRG4 (nebo jeho ekvivalent v příkazovém řádku STRMQMTRM, podrobnosti viz Spuštění MQ Monitor spouštěčů (STRMQMTRM)). místo AMQSERV4 pro odeslání úlohy místo toho, ale potenciální prodlevy při odesílání úlohy by mohly zlehčovat sledování toho, co se děje.
- Spusťte soubor SYSTEM.SAMPLE.INQUIRE a ukázkové programy SYSTEM.SAMPLE.SET . Pomocí ukázkového datového souboru jsou příkazy pro vydání požadavků programu C na tyto servery následující:

```
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 vzorového programu požadavku
Program otevře frontu cílového serveru, aby mohl vkládat zprávy. Používá volání MQOPEN s volbou MQOO_OUTPUT. Pokud nemůže otevřít frontu, program zobrazí chybovou zprávu obsahující kód příčiny vrácený voláním MQOPEN.

Program poté otevře frontu pro odpověď s názvem SYSTEM.SAMPLE.REPLY , aby mohla získat zprávy odpovědi. K tomu program používá volání MQOPEN s volbou MQOO_INPUT_EXCLUSIVE. Pokud nemůže otevřít frontu, program zobrazí chybovou zprávu obsahující kód příčiny vrácený voláním MQOPEN.

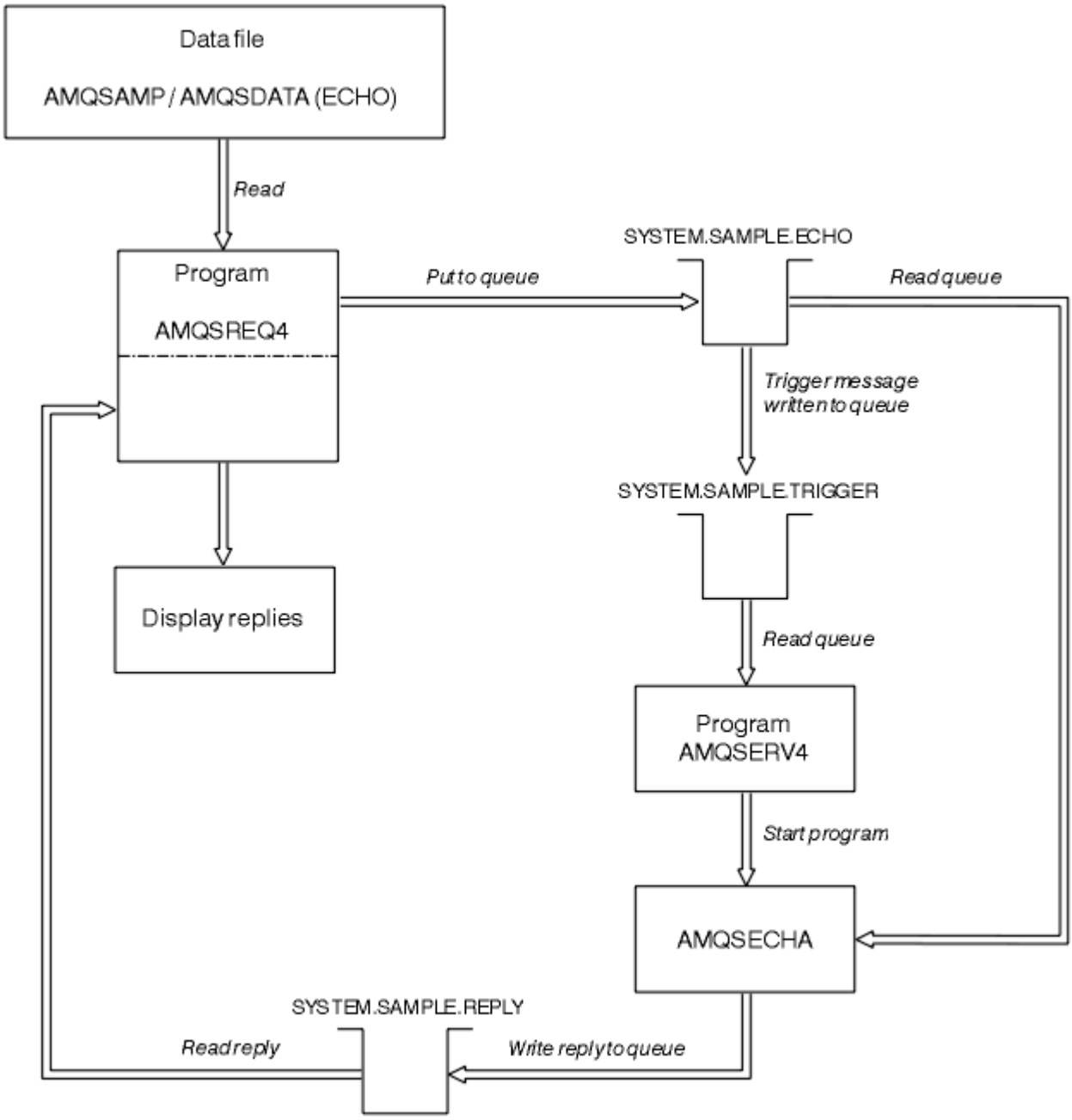
Pro každý řádek vstupu pak program načte text do vyrovnávací paměti a použije volání MQPUT k vytvoření zprávy požadavku obsahující text tohoto řádku. Při tomto volání program používá volbu sestavy MQRO_EXCEPTION_WITH_DATA k požadavku, aby všechny zprávy sestavy odeslané o zprávě požadavku obsahovaly prvních 100 bajtů dat zprávy. Program pokračuje, dokud nedosáhne konce vstupu nebo dokud neselže volání MQPUT.

Program poté pomocí volání MQGET odebere zprávy 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 COBOL a 1 minuta ve verzi C pro první odpověď (aby byl čas na spuštění serverové aplikace) a 15 sekund pro následné odpovědi. Program čeká na tato období, pokud ve frontě není žádná zpráva. Pokud před vypršením tohoto intervalu nepřijde žádná zpráva, volání selže a vrátí kód příčiny MQRC_NO_MSG_AVAILABLE. Volání také používá volbu MQGMO_ACCEPT_TRUNCATED_MSG, aby byly zprávy delší než deklarovaná velikost vyrovnávací paměti oříznuty.

Program demonstruje, jak vymazat pole *MsgId* a *CorrelId* struktury MQMD po každém volání MQGET, protože volání nastaví tato pole na hodnoty obsažené ve zprávě, kterou načte. Vymazání těchto polí znamená, že následná volání MQGET načítají zprávy v pořadí, ve kterém jsou zprávy zadrženy ve frontě.

Program pokračuje, dokud volání MQGET nevrátí kód příčiny MQRC_NO_MSG_AVAILABLE nebo dokud volání MQGET neselže. Pokud volání selže, program zobrazí chybovou zprávu, která obsahuje kód příčiny.

Program poté zavře frontu cílového serveru i frontu pro odpověď pomocí volání MQCLOSE.



Obrázek 133. Ukázkový graf toku programu klienta/serveru IBM i (Echo)

Ukázkové programy Set

Ukázkové programy Set blokují operace vložení do fronty pomocí volání MQSET pro změnu atributu **InhibitPut** fronty. Také se dozvíte o návrhu ukázkových programů Set.

Názvy těchto programů viz “Funkce demonstrované v ukázkových programech na platformě Multiplatforms” na stránce 1025 .

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 zjišťovány. Verze C také používá název správce front. Verze COBOL používá výchozího správce front.

Aby spouštěcí proces fungoval, ujistěte se, že ukázkový program Set, který chcete použít, je spuštěn zprávami přicházejícími do fronty SYSTEM.SAMPLE.SET. Chcete-li tak učinit, zadejte název ukázkového programu Set, který chcete použít, do pole *ApplicId* definice procesu SYSTEM.SAMPLE.SETPROCESS.

Ukázková fronta má typ spouštěče FIRST. Pokud již existují zprávy ve frontě před spuštěním ukázky požadavku, není ukázka sady spuštěna odesílanými zprávami.

Po správném nastavení definice:

- ALW V systémech AIX, Linux, and Windows spusťte program **runmqtrm** v jedné relaci a poté spusťte program amqsreq v jiné relaci.
- IBM i Pro systém IBM ispusťte program AMQSERV4 v jedné relaci, pak spusťte program AMQSREQ4 v jiné relaci. Můžete použít AMQSTRG4 místo AMQSERV4, ale potenciální prodlevy při odesílání úloh by mohly způsobit, že bude méně snadné sledovat, co se děje.

Ukázkové programy požadavků použijte k odeslání zpráv požadavků, z nichž každá obsahuje pouze název fronty, do fronty SYSTEM.SAMPLE.SET. Pro každou zprávu požadavku odešlou vzorové programy zprávu odpovědi obsahující potvrzení, že operace vložení byly v uvedené frontě zablokovány. Odpovědi se odešlou do fronty pro odpověď uvedené ve zprávě požadavku.

Návrh vzorového programu Set

Program otevře frontu uvedenou ve struktuře zprávy spouštěče, která byla předána při jejím spuštění. (Pro přehlednost toto nazýváme *frontou požadavků*.) Program používá volání MQOPEN k otevření této fronty pro sdílený vstup.

Program používá volání MQGET k odebrání 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, aby zjistil, zda se jedná o zprávu požadavku; pokud není, program zprávu vyřadí a zobrazí varovnou zprávu.

Pro každou zprávu požadavku odebranou z fronty požadavků program přečte název fronty (kterou budeme volat *cílovou frontu*). obsažené v datech a otevře tuto frontu pomocí volání MQOPEN s volbou MQOO_SET. Program poté pomocí volání MQSET nastaví hodnotu atributu **InhibitPut** cílové fronty na hodnotu MQQA_PUT_INHIBITED.

Je-li volání MQSET úspěšné, program pomocí volání MQPUT1 vloží zprávu odpovědi do fronty pro odpověď. Tato zpráva obsahuje řetězec PUT `inhibited`.

Pokud je volání MQOPEN nebo MQSET neúspěšné, program použije volání MQPUT1 k vložení zprávy `report` do fronty pro odpověď. V poli *Feedback* deskriptoru této zprávy sestavy je uveden kód příčiny vrácený voláním MQOPEN nebo MQSET v závislosti na tom, který z nich se nezdařil.

Po volání MQSET program zavře cílovou frontu pomocí volání MQCLOSE.

Pokud ve frontě požadavků nezbývají žádné zprávy, program tuto frontu zavře a odpojí se od správce front.

Ukázkový program TLS

AMQSSSLC je ukázkový program jazyka C, který demonstruje, jak používat struktury MQCNO a MQSCO k dodání informací o připojení klienta TLS ve volání MQCONNX. To umožňuje aplikaci MQI klienta poskytnout definici svého kanálu připojení klienta a nastavení TLS za běhu bez tabulky CCDT (Client Channel Definition Table).

Je-li zadán název připojení, program vytvoří definici kanálu připojení klienta ve struktuře MQCD.

Je-li dodán název kmene souboru úložiště klíčů, program vytvoří strukturu MQSCO; je-li dodán také odpovídací modul OCSP URL , program vytvoří strukturu MQAIR záznamu ověřovacích informací.

Program se poté připojí ke správci front pomocí MQCONNX. Zjišťuje a vypisuje název správce front, ke kterému je připojen.

Tento program je určen k propojení jako aplikace klienta MQI. Lze jej však propojit jako běžnou aplikaci MQI. V takovém případě se pouze připojí k lokálnímu správci front a ignoruje informace o připojení klienta.

V 9.3.0 V 9.3.0 Není-li přístupová fráze pro přístup k úložišti klíčů ukládána do souboru, musíte tuto přístupovou frázi zadat do souboru **amqsssl c** při spuštění aplikace. Přístupovou frázi můžete zadat buď:

- Požadujete-li výzvu k zadání hesla pro **amqssslc** , nebo
- Pomocí proměnné prostředí *MQKEYRPWD* nebo
- Použití atributu **SSLKeyRepositoryPassword** v konfiguračním souboru klienta

Další informace o zadání hesla úložiště klíčů do aplikací IBM MQ MQI client naleznete v tématu Dodání hesla úložiště klíčů pro IBM MQ MQI client on AIX, Linux, and Windows.

Produkt **amqssslc** přijímá následující parametry, které jsou všechny 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:

V 9.3.0 V 9.3.0 **-k KeyReposFileName**
Název souboru úložiště klíčů. Není-li přípona souboru dodána, předpokládá se, že je .kdb. Příklad:

```
/home/user/client.kdb
C:\User\client.p12
```

-s CipherSpec
Řetězec CipherSpec kanálu TLS odpovídající hodnotě **SSLCIPH** v definici kanálu SVRCONN ve správci front.

-f
Uvádí, že se musí použít pouze certifikované algoritmy FIPS 140-2.

-b VALUE1[,VALUE2...]
Určuje, že musí být použity pouze algoritmy vyhovující standardu Suite B. Tento parametr je čárkami oddělený seznam jedné nebo více následujících hodnot: NONE,128_BIT,192_BIT. Tyto hodnoty mají stejný význam jako pro proměnnou prostředí **MQSUITEB** a ekvivalentní nastavení **EncryptionPolicySuiteB** v sekci SSL konfiguračního souboru klienta.

-p Zásada
Uvádí zásadu ověření platnosti certifikátu, která se má použít. Může se jednat o jednu z následujících hodnot:

ANY
Použijte všechny zásady ověřování certifikátů podporované knihovnou zabezpečených soketů a přijměte řetěz certifikátů, pokud některá ze zásad považuje řetěz certifikátů za platný. Toto nastavení lze použít pro maximální zpětnou kompatibilitu se staršími digitálními certifikáty, které nevyhovují moderním certifikačním normám.

RFC5280
Použijte pouze zásadu ověření certifikátu vyhovující standardu RFC 5280. Toto nastavení poskytuje přísnější ověření než nastavení ANY, ale odmítá některé starší digitální certifikáty.

Výchozí hodnota je ANY.

-l CertLabel
Popisek certifikátu, který se má použít pro zabezpečené připojení.

Poznámka: Hodnotu musíte zadat pomocí malých písmen.

V 9.3.0 V 9.3.0 **-w**
Určuje, že produkt **amqssslc** vyzve k zadání přístupové fráze úložiště klíčů, která má být dodána.

V 9.3.0 V 9.3.0 **-i**

Určuje, že produkt **amqssslc** vyzve k zadání počátečního klíče použitého k zašifrování přístupové fráze úložiště klíčů, která má být dodána.

Tuto volbu určete, pokud byl při šifrování přístupové fráze úložiště klíčů pomocí obslužného programu **runmqicred** zadán počáteční soubor s klíči.

Parametr odvolání certifikátu OCSP:

-o URL
URL odpovídacího modulu OCSP.

Můžete také nastavit jednu z následujících proměnných prostředí pro dodání pověření, která se používají k ověření u správce front:

MQSAMP_USER_ID
Nastavte na ID uživatele, které se má použít pro ověření připojení, pokud chcete použít ID uživatele a heslo pro ověření u správce front. Program vyzve k zadání hesla, které má doprovázet ID uživatele.

Linux V 9.3.4 AIX **MQSAMP_TOKEN**
Nastavte na neprázdnou hodnotu, chcete-li zadat token ověření pro ověření u správce front. Program vyzve k zadání tokenu ověření.

Spuštění ukázkového programu TLS
Chcete-li spustit ukázkový program TLS, musíte nejprve nastavit prostředí TLS. Poté spustíte ukázku z příkazového řádku a zadáte řadu parametrů.

Informace o této úloze

Následující pokyny spustí ukázkový program pomocí osobních certifikátů. Změnou příkazu můžete například použít certifikáty CA a zkontrolovat jejich stav pomocí odpovídacího modulu OCSP. Viz pokyny v ukázce.

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 produktu AIX, Linux, and Windows.
3. Vytvořte úložiště klíčů pro klienta. Nazvějte jej *clientkey.kdb*.

   Při vytváření úložiště klíčů uložte heslo úložiště klíčů do souboru.
4. Vytvořte osobní certifikát pro správce front. Další informace naleznete v tématu Vytvoření osobního certifikátu podepsaného držitelem na webu AIX, 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 Extrahování veřejné části certifikátu podepsaného (svým) držitelem z úložiště klíčů v systému AIX, Linux, and Windowsa Přidání certifikátu CA (nebo veřejné části certifikátu podepsaného (svým) držitelem) do úložiště klíčů v systému AIX, Linux, and Windows.
7. Extrahujte osobní certifikát z úložiště klíčů klienta a přidejte jej do úložiště klíčů serveru.
8. Vytvořte kanál připojení serveru pomocí příkazu MQSC:

   ```
   DEFINE CHANNEL(QM1SVRCONN) CHLTYPE(SVRCONN) TRPTYPE(TCP)
   SSLCIPH(TLS_RSA_WITH_AES_128_CBC_SHA256)
   ```

   Další informace naleznete v tématu Kanál připojení serveru .
9. Definujte a spusťte modul listener kanálu ve správci front. Další informace viz DEFINE LISTENER a START LISTENER.
10. Spusťte ukázkový program pomocí následujícího příkazu:

V 9.3.0 V 9.3.0

```
AMQSSSLC -m QM1 -c QM1SVRCONN -x localhost
-k "C:\Program Files\IBM\MQ\clientkey.kdb" -s TLS_RSA_WITH_AES_128_CBC_SHA256
-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 zjišťuje jeho název.
3. Zavře správce front.
4. Odpojí se od správce front.

Pokud se ukázkový program úspěšně spustí, 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_SHA256
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 URLodpovídacího modulu OCSP, obdržíte následující zprávu:

```
MQCONNX ended with reason code 2553
```

Seznam kódů příčiny viz Kódy příčiny a dokončení rozhraní API.

Ukázkové programy spuštění

Funkce poskytnutá v ukázce spouštěče je podmnožinou funkce poskytnuté v monitoru spouštěčů v programu **runmqtrm** .

Názvy těchto programů viz “Funkce demonstrované v ukázkových programech na platformě Multiplatforms” na stránce 1025 .

Návrh spouštějícího vzorku

Ukázkový program, který spustil, otevře inicializační frontu pomocí volání MQOPEN s volbou MQOOO_INPUT_AS_Q_DEF. Získává zprávy z inicializační fronty pomocí volání MQGET s volbami MQGMO_ACCEPT_TRUNCATED_MSG a MQGMO_WAIT, které určují 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 zprávu otestuje kontrolou velikosti zprávy, aby se ujistil, že má stejnou velikost jako struktura MQTM. Pokud tento test selže, program zobrazí varování.

V případě platných zpráv spouštěče ukázka spouštěče zkopíruje data z těchto polí: *ApplicId*, *EnvrData*, *Versiona* *ApplType*. Poslední dvě z těchto polí jsou číselná, takže program vytvoří náhrady znaků, které se použijí ve struktuře MQTMC2 pro systémy IBM i, AIX, Linux, and Windows .

Ukázka spouštěče vydá spouštěcí příkaz pro aplikaci určenou v poli *ApplicId* zprávy spouštěče a předá strukturu MQTMC2 nebo MQTMC (znaková verze zprávy spouštěče).

- ALW V systémech AIX, Linux, and Windows se pole *EnvData* používá jako rozšíření vyvolávajícího příkazového řetězce.
- IBM i V produktu IBM ise používá jako parametry pro zadání úlohy, například priorita úlohy nebo popis úlohy.

Nakonec program uzavře inicializační frontu.

Ukončení spouštění ukázkových programů na systému IBM i

IBM i

Program monitoru spouštěčů lze ukončit pomocí volby sysrequest 2 (ENDRQS) nebo blokováním získání z fronty spouštěčů.

Pokud se použije vzorová fronta spouštěčů, příkaz je:

```
CHGMQMQ QNAME('SYSTEM.SAMPLE.TRIGGER') MQMNAME GETENBL(*NO)
```

Důležité: Před opětovným spuštěním v této frontě musíte zadat následující příkaz:

```
CHGMQMQ QNAME('SYSTEM.SAMPLE.TRIGGER') GETENBL(*YES)
```

Spuštění ukázkových programů Triggering
Toto téma obsahuje informace o spouštění ukázkových programů spouštěče.

Spuštění ukázek amqstrg0.c, amqstrg a amqstrgc

Program má 2 parametry:

1. Název inicializační fronty (nezbytné)
2. Název správce front (volitelné)

Není-li určen správce front, připojí se k výchozímu správci front. Při spuštění příkazu amqscos0.tst; bude definována vzorová inicializační fronta. Název této fronty je SYSTEM.SAMPLE.TRIGGERa můžete jej použít při spuštění tohoto programu.

Poznámka: Funkce v této ukázce je podmnožinou úplné spouštěcí funkce, která je dodávána v programu runmqtrm.

Spuštění ukázky AMQSTRG4

IBM i

Jedná se o monitor spouštěčů pro prostředí IBM i . Odešle jednu úlohu IBM i pro každou aplikaci, která se má spustit. To znamená, že ke každé zprávě spouštěče je přidruženo další zpracování.

AMQSTRG4 (v QCSRC) má dva parametry: název inicializační fronty, kterou má obsluhovat, a název správce front (volitelné). AMQSAMP4 (v QCLSRC) definuje vzorovou inicializační frontu SYSTEM.SAMPLE.TRIGGER, který můžete použít při pokusu o ukázkové programy.

Pomocí vzorové fronty spouštěčů je příkaz, který se má vydat:

```
CALL PGM(QMQM/AMQSTRG4) PARM('SYSTEM.SAMPLE.TRIGGER')
```

Alternativně můžete použít ekvivalent CL STRMQMTRM; podrobnosti viz Spuštění MQ Monitor spouštěče (STRMQMTRM).

Spuštění ukázky AMQSERV4

IBM i

Toto je spouštěcí server pro prostředí IBM i . Pro každou zprávu spouštěče tento server spustí příkaz spuštění ve své vlastní úloze, aby spustil uvedenou aplikaci. Spouštěcí server může volat transakce CICS .

AMQSERV4 má dva parametry: název inicializační fronty, kterou má obsluhovat, a název správce front (volitelné). AMQSAMP4 definuje inicializační frontu vzorku, SYSTEM.SAMPLE.TRIGGER, který můžete použít při pokusu o ukázkové programy.

Pomocí vzorové fronty spouštěčů je příkaz, který se má vydat, následující:

```
CALL PGM(QMQM/AMQSERV4) PARM('SYSTEM.SAMPLE.TRIGGER')
```

Návrh spouštěcího serveru
Návrh spouštěcího serveru je podobný jako u monitoru spouštěčů, s několika výjimkami

Návrh spouštěcího serveru je podobný jako u monitoru spouštěčů, kromě toho, že spouštěcí server:

- Umožňuje MQAT_CICS i aplikace MQAT_OS400 .
- IBM i Volá aplikace IBM i ve své vlastní úloze (nebo používá příkaz STRCICSUSR ke spuštění aplikací CICS), místo aby zadal úlohu IBM i .
- V případě aplikací CICS nahradí řetězec *EnvData*, například uvedením oblasti CICS , ze zprávy spouštěče v příkazu STRCICSUSR.
- Otevře inicializační frontu pro sdílený vstup, aby mohlo být současně spuštěno mnoho spouštěcích serverů.

Poznámka: Programy spuštěné pomocí AMQSERV4 nesmí používat volání MQDISC, protože tím se zastaví spouštěcí server. Pokud programy spuštěné AMQSERV4 používají volání MQCONN, získají kód příčiny MQRC_ALREADY_CONNECTED.

ALW *Použití ukázek TUXEDO v systému AIX, Linux, and Windows*

Seznamte se s ukázkovými programy Put a Get pro TUXEDO a sestavte prostředí serveru v TUXEDO.

Než začnete

Před spuštěním těchto ukázek musíte sestavit prostředí serveru.

Informace o této úloze

Poznámka: V této části se znak zpětného lomítka (\) používá k rozdělení dlouhých příkazů na více než jeden řádek. Tento znak nezadávejte. Zadejte každý příkaz jako jeden řádek.

ALW *Sestavení prostředí serveru*
Informace o sestavení prostředí serveru pro produkt IBM MQ pro různé platformy.

Než začnete

Předpokládá se, že máte funkční prostředí TUXEDO.

AIX *Sestavení prostředí serveru pro AIX (32bitové)*
Jak sestavit prostředí serveru pro IBM MQ for AIX (32bitové).

Postup

1. Vytvořte adresář (například APPDIR), ve kterém je sestaveno 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. Do souboru TUXEDO udataobj/RMpřidejte následující řádek:

```
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 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 tuto výzvu zadejte:

```
> crdl -z /APPDIR/TLOG1
```

7. Spusťte správce front:

```
$ strmqm
```

8. Začátek 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 AIX (64bitové)
Jak sestavit prostředí serveru pro IBM MQ for AIX (64bitové).

Postup

1. Vytvořte adresář (například APPDIR), ve kterém je sestaveno 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. Do souboru TUXEDO udataobj/RMpřidejte následující řádek:

```
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 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 tuto výzvu zadejte:

```
> crdl -z /APPDIR/TLOG1
```

7. Spusťte správce front:

```
$ strmqm
```

8. Začátek 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 Windows (32bitové)*
Sestavení prostředí serveru pro IBM MQ for Windows (32bitové).

Informace o této úloze

Poznámka: Změňte pole označená jako *PROMĚNNÉ* v následujících cestách k adresářům:

Tabulka 164. Pole, která se mají změnit na cesty k adresářům	
Pole	**Cesta k adresáři**
MQMDIR	Cesta k adresáři určená při instalaci produktu IBM MQ , například `g:\Program Files\IBM\MQ`.
TUXDIR	Cesta k adresáři určená při instalaci produktu TUXEDO, například `f:\tuxedo`.
APPDIR	Cesta k adresáři, který se má použít 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 134. Příklad souboru ubbstxcn.cfg pro IBM MQ for Windows

Poznámka: Změňte název počítače *MachineName*a cesty k adresářům tak, aby odpovídaly vaší instalaci. Změňte také název správce front *MYQUEUEMANAGER* na název správce front, ke kterému se chcete připojit.

Ukázkový soubor `ubbconfig` pro soubor IBM MQ for Windows je uveden v seznamu Obrázek 134 na stránce 1090. Dodává se jako `ubbstxcn.cfg` v adresáři ukázek IBM MQ .

Ukázkový soubor Makefile (viz Obrázek 135 na stránce 1091) dodaný pro IBM MQ for Windows se nazývá `ubbstxmn.maka` je uložen 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 135. Ukázkový soubor makefile TUXEDO pro IBM MQ for Windows

Chcete-li sestavit prostředí serveru a ukázky, postupujte takto.

Postup

1. Vytvořte adresář aplikace, ve kterém 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, abyste nastavili názvy adresářů a cesty k adresářům použité ve vaší instalaci.
4. Upravte soubor `ubbstxcn.cfg` (viz Obrázek 134 na stránce 1090) a přidejte podrobnosti o názvu počítače a správci front, ke kterému se chcete připojit.
5. Do souboru TUXEDO *TUXDIR*`udataobj\rm`přidejte následující řádek:

   ```
   MQSERIES_XA_RMI;MQRMIXASwitchDynamic;MQMDIR\tools\lib\mqmxa.lib MQMDIR\tools\lib\mqm.lib
   ```

   Nová položka musí být v souboru jeden řádek.
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 `tmadmin -c`a zadejte následující příkaz:

   ```
   crdl -z APPDIR\TLOG
   ```

8. Nastavte aktuální adresář na *APPDIR*a vyvolejte ukázkový soubor Makefile `amqstxmn.mak` jako externí soubor Makefile projektu. Například s produktem Microsoft Visual C++ zadejte následující příkaz:

   ```
   msvc amqstxmn.mak
   ```

   Chcete-li sestavit všechny ukázkové programy, vyberte volbu **sestavit** .

Windows *Sestavení prostředí serveru pro Windows (64bitové)*
Jak sestavit prostředí serveru pro IBM MQ for Windows (64bitové).

Informace o této úloze

Poznámka: Změňte pole označená jako *PROMĚNNÉ* v následujících cestách k adresářům:

Tabulka 165. Pole, která se mají změnit na cesty k adresářům

Pole	**Cesta k adresáři**
MQMDIR	Cesta k adresáři určená při instalaci produktu IBM MQ , například `g:\Program Files\IBM\MQ`.
TUXDIR	Cesta k adresáři určená při instalaci produktu TUXEDO, například `f:\tuxedo`.
APPDIR	Cesta k adresáři, který se má použít 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 136. Příklad souboru ubbstxcn.cfg pro IBM MQ for Windows

Poznámka: Změňte název počítače *MachineName*a cesty k adresářům tak, aby odpovídaly vaší instalaci. Změňte také název správce front *MYQUEUEMANAGER* na název správce front, ke kterému se chcete připojit.

Ukázkový soubor ubbconfig pro IBM MQ for Windows je uveden v seznamu Obrázek 136 na stránce 1093. Dodává se jako `ubbstxcn.cfg` v adresáři ukázek IBM MQ .

Ukázkový soubor Makefile (viz Obrázek 137 na stránce 1094) pro IBM MQ for Windows se nazývá `ubbstxmn.maka` je uložen 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 137. Ukázkový soubor makefile TUXEDO pro IBM MQ for Windows

Chcete-li sestavit prostředí serveru a ukázky, postupujte takto.

Postup

1. Vytvořte adresář aplikace, ve kterém 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, abyste nastavili názvy adresářů a cesty k adresářům použité ve vaší instalaci.
4. Upravit `ubbstxcn.cfg` (viz Obrázek 136 na stránce 1093) 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
   ```

   Nová položka musí být v souboru jeden řádek.
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 `tmadmin -ca` zadejte příkaz:

   ```
   crdl -z APPDIR\TLOG
   ```

8. Nastavte aktuální adresář na *APPDIR*a vyvolejte ukázkový soubor Makefile `amqstxmn.mak` jako externí soubor Makefile projektu. Například s produktem Microsoft Visual C++ zadejte následující příkaz:

```
msvc amqstxmn.mak
```

   Chcete-li sestavit všechny ukázkové programy, vyberte volbu **sestavit** .

ALW *Ukázkový program serveru pro TUXEDO*

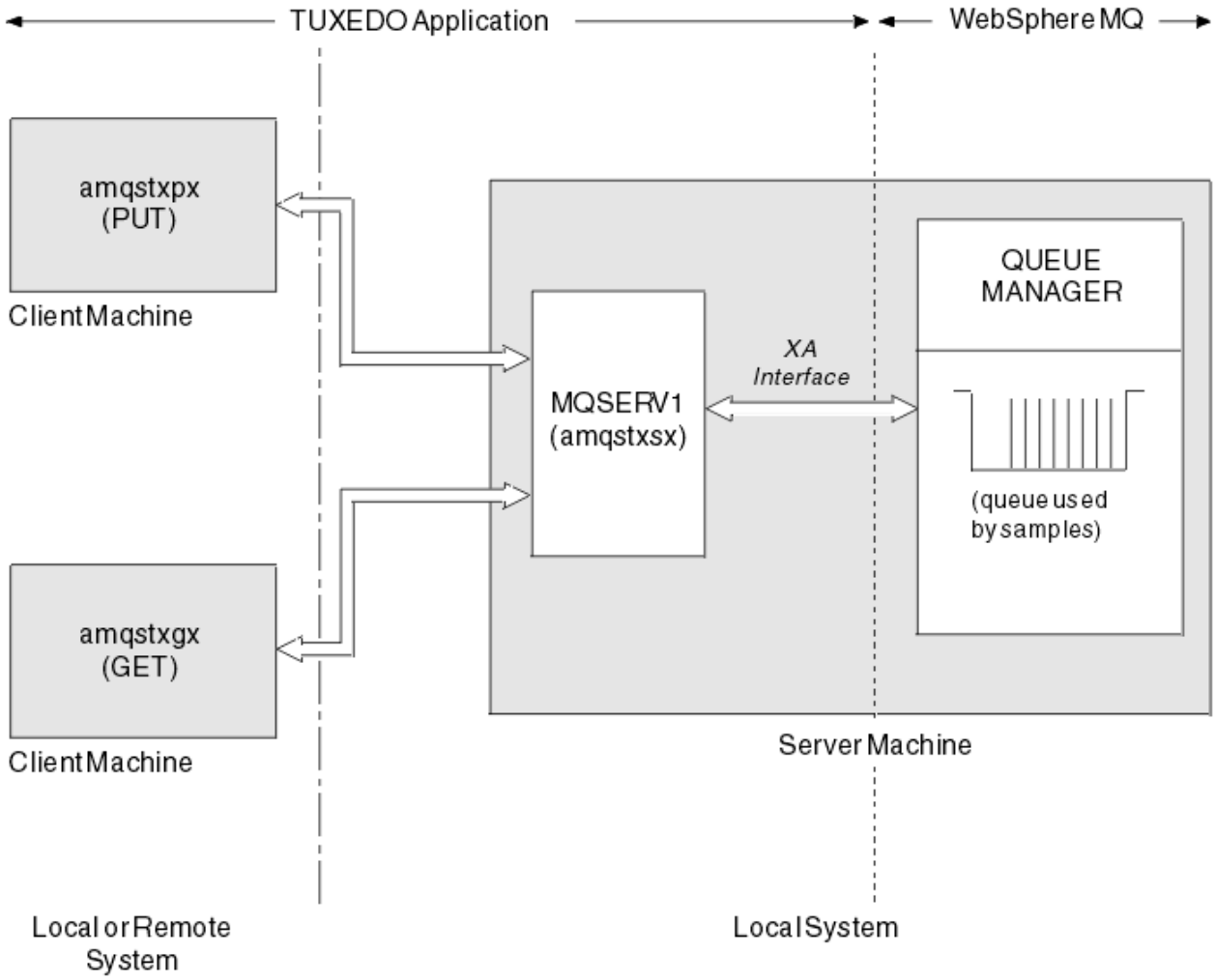
Ukázkový program serveru (amqstxsx) je navržen pro spuštění s ukázkovými programy Put (amqstxpx.c) a Get (amqstxgx.c). Ukázkový program serveru se spustí automaticky při spuštění TUXEDO.

Poznámka: Před spuštěním produktu TUXEDO je třeba spustit správce front.

Ukázkový server poskytuje dvě služby TUXEDO, MPUT1 a MGET1:

- Služba MPUT1 je řízena ukázkou PUT a používá MQPUT1 v synchronizačním bodu k vložení zprávy do pracovní jednotky řízené TUXEDO. Vezme parametry QName a Text zprávy, které jsou dodány ukázkou PUT.
- Služba MGET1 otevře a zavře frontu pokaždé, když obdrží 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 138. Jak vzorky TUXEDO spolupracují

ALW *Vložit ukázkový program pro TUXEDO*

Tato ukázka vám umožňuje vložit zprávu do fronty vícekrát, v dávkách, a demonstrovat tak synchronizaci s použitím TUXEDO jako správce prostředků.

Ukázkový program serveru `amqstxsx` musí být spuštěn, aby byla ukázka vložení úspěšná; 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"`

Tím se do fronty s názvem `myqueue`vloží 30 zpráv v šesti dávkách, z nichž každá obsahuje pět zpráv. Pokud dojde k problémům, vrací dávku zpráv zpět, jinak je potvrdí.

Všechny chybové zprávy jsou zapsány do souboru protokolu TUXEDO a do souboru stderr. Všechny kódy příčiny jsou zapsány do stderr.

ALW *Získat vzorek pro TUXEDO*

Tato ukázka vám umožňuje získat zprávy z fronty v dávkách.

Aby byla ukázka Get úspěšná, musí být spuštěn program ukázkového serveru `amqstxsx` ; program ukázkového 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 vezme 24 zpráv z fronty s názvem `myqueue`, v šesti dávkách, z nichž každá obsahuje čtyři zprávy. Spustíte-li toto po příkladu vložení, který vloží 30 zpráv na systém `myqueue`, máte pouze šest zpráv na systému `myqueue`. Počet dávek a velikost dávky se mohou lišit mezi vložením zpráv a jejich získáním.

Všechny chybové zprávy jsou zapsány do souboru protokolu TUXEDO a do souboru stderr. Všechny kódy příčiny jsou zapsány do stderr.

Windows *Použití uživatelské procedury zabezpečení rozhraní SSPI v systému Windows*

Toto téma popisuje, jak používat programy uživatelské procedury kanálu 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 součást produktu IBM MQ for Windows ve složce *MQ_INSTALLATION_PATH*`/exits/` *INSTALLATION_NAME* . Například `C:\Program Files\IBM\MQ\exits\installation2`. Načte se jako standardní uživatelská procedura. Můžete spustit zadanou uživatelskou proceduru kanálu zabezpečení 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 v kanálu SVRCONN následující:

```
SCYDATA('remote_principal_name')
```

kde *remote_principal_name* je ve formátu *DOMAIN\uživatel*. Zabezpečený kanál je zaveden pouze v případě, že název vzdáleného činitele odpovídá hodnotě *remote_principal_name*.

Chcete-li použít dodané programy uživatelské procedury kanálu mezi systémy, které pracují v rámci domény zabezpečení Kerberos , vytvořte pro správce front **servicePrincipalName** .

Zdrojový kód

Soubor se zdrojovým kódem ukončení se nazývá `amqsspin.c`. Nachází se v adresáři `C:\Program Files\IBM\MQ\Tools\c\Samples`.

Pokud upravíte zdrojový kód, musíte znovu zkompilovat upravený zdroj.

Kompilujte ji a propojte ji stejným způsobem jako jakoukoli jinou uživatelskou proceduru kanálu pro příslušnou platformu, s výjimkou toho, že k záhlavím SSPI je třeba přistupovat v době kompilace a knihovny zabezpečení SSPI spolu s doporučenými přidruženými knihovnami je třeba přistupovat v době propojení.

Před provedením následujícího příkazu se ujistěte, že jsou ve vaší cestě k dispozici soubory `cl.exe`, knihovna Visual C++ a složka `include` . 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 nezahrnuje žádné ustanovení pro trasování nebo ošetření chyb. Pokud upravíte a použijete zdrojový kód, přidejte vlastní rutiny pro trasování a ošetřování chyb.

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 ukázkovou definici, změňte hodnotu OTHER na název druhého správce front, kterého chcete použít. Musíte také nastavit kanál zpráv mezi dvěma správci front. Informace o tom, jak to provést, naleznete v tématu Definování kanálů.

Ukázkové programy požadavků vložily svůj vlastní název lokálního správce front do pole *ReplyToQMgr* zpráv, které odesílají. Ukázky inquire a Set odesílají zprávy odpovědí do fronty a správce front zpráv uvedeného v polích *ReplyToQ* a *ReplyToQMgr* zpráv požadavků, které zpracovávají.

Ukázkový program pro monitorování front klastru (AMQSCLM)

Tato ukázka používá vestavěné funkce vyrovnávání pracovní zátěže klastru IBM MQ ke směrování zpráv do instancí front, k nimž jsou připojeny spotřební aplikace. Tento automatický směr zabraňuje vytváření zpráv v instanci fronty klastru, ke které není připojena žádná aplikace, která spotřebovává.

Přehled

Můžete nastavit klastr, který má více než jednu definici pro stejnou frontu v různých správcích front. Tato konfigurace poskytuje výhodu zvýšené dostupnosti a vyvažování pracovní zátěže. Do produktu IBM MQ však není vestavěna žádná schopnost dynamicky upravovat distribuci zpráv v rámci klastru na základě stavu připojených aplikací. Z tohoto důvodu musí být aplikace odběratele vždy připojena ke každé instanci fronty, aby se zajistilo zpracování zpráv.

Ukázkový program monitorování fronty klastru monitoruje stav připojených aplikací. Program dynamicky upravuje vestavěnou konfiguraci vyrovnávání pracovní zátěže tak, aby směrovala zprávy na instance klastrované fronty s připojenými spotřebními aplikacemi. V určitých situacích lze tento program použít k uvolnění potřeby, aby aplikace, která spotřebovává, byla vždy připojena ke každé instanci fronty. Také znovu odešle zprávy, které se zařadí do fronty v instanci fronty bez připojených spotřebních aplikací. Opakované odesílání zpráv umožňuje, aby byly zprávy směrovány kolem přijímající aplikace, která je dočasně vypnuta.

Program je navržen tak, aby byl používán tam, kde jsou aplikace, které spotřebovávají, aplikacemi s dlouhou dobou zpracování, spíše než aby často připojovaly a odpojovaly aplikace.

Ukázkový program monitorování fronty klastru je kompilovaný spustitelný program ukázkového 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 funguje ukázkový program monitorování fronty klastru, body, které je třeba vzít v úvahu při nastavení systému pro spuštění ukázkového programu, a úpravy, které lze provést v ukázkovém zdrojovém kódu.

Návrh

Ukázkový program monitorování fronty klastru monitoruje lokální klastrované fronty, které mají připojené aplikace, které spotřebovávají. Program monitoruje fronty uvedené 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, jež má být monitorována, vyžaduje, aby k němu byla připojena instance ukázkového programu monitorování fronty klastru.

Dynamické směrování zpráv

Ukázkový program monitorování fronty klastru používá hodnotu **IPPROCS** (otevřít pro počet vstupních procesů) fronty k určení, zda má tato fronta nějaké spotřebitele. Hodnota větší než 0 označuje, že fronta má připojenou alespoň jednu aplikaci, která spotřebovává. Tyto fronty jsou aktivní. Hodnota 0 označuje, že fronta nemá žádné připojené programy, které by spotřebovávalo. Tyto fronty jsou neaktivní.

Pro klastrovanou frontu s více instancemi v klastru IBM MQ používá vlastnost priority pracovní zátěže klastru **CLWLPRTY** každé instance fronty k určení, do kterých instancí se mají odesílat zprávy. Produkt IBM MQ odesílá zprávy do dostupných instancí fronty s nejvyšší hodnotou **CLWLPRTY** .

Ukázkový program monitorování fronty klastru aktivuje frontu klastru nastavením lokální hodnoty **CLWLPRTY** na hodnotu 1. Program deaktivuje frontu klastru nastavením její hodnoty **CLWLPRTY** na 0.

IBM MQ technologie klastrování šíří aktualizovanou vlastnost **CLWLPRTY** klastrované fronty do všech příslušných správců 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 klastru.

Přenos zpráv ve frontě

Dynamická úprava hodnoty **CLWLPRTY** ovlivňuje směrování nových zpráv. Tato dynamická úprava nemá vliv na zprávy, které již byly zařazeny do fronty v instanci fronty bez připojených spotřebitelů, nebo na zprávy, které byly prostřednictvím mechanizmu vyrovnávání pracovní zátěže před šířením upravené hodnoty **CLWLPRTY** v rámci klastru. Výsledkem je, že zprávy zůstanou v jakékoli neaktivní frontě a nebudou zpracovány přijímající aplikací. K vyřešení tohoto problému je ukázkový program monitorování fronty klastru schopen získat zprávy z lokální fronty bez odběratelů a odeslat tyto zprávy vzdáleným instancím stejné fronty, kde jsou odběratelé připojeni.

Ukázkový program monitorování fronty klastru přenáší zprávy z neaktivní lokální fronty do jedné nebo více aktivních vzdálených front získáním zpráv (pomocí **MQGET**). a vkládání zpráv (pomocí **MQPUT**) do stejné klastrované fronty. Tento přenos způsobí, že správa pracovní zátěže klastru IBM MQ vybere jinou cílovou

instanci na základě vyšší hodnoty CLWLPRTY , než je hodnota lokální instance fronty. Perzistence zpráv a kontext jsou během přenosu zpráv zachovány. Pořadí zpráv a žádné volby vazby nejsou zachovány.

Naplánování

Ukázkový program monitorování fronty klastru upravuje konfiguraci klastru, když dojde ke změně konektivity spotřebních aplikací. Úpravy jsou přeneseny ze správců front, kde ukázkový program monitorování front klastru monitoruje fronty, do správců front úplného úložiště v klastru. Správci front úplného úložiště zpracují aktualizace konfigurace a znovu je odešlete všem příslušným správcům front v klastru. Mezi relevantní správce front patří správci front, kteří vlastní klastrované fronty se stejným názvem (kde je spuštěna instance ukázkového programu monitorování front klastru), a všichni správci front, kde aplikace otevřela frontu klastru, aby do ní za posledních 30 dnů vložila zprávy.

Změny jsou asynchronně zpracovány v rámci klastru. Proto po každé změně mohou mít různí správci front v klastru různé pohledy na konfiguraci po určitou dobu.

Ukázkový program pro monitorování fronty klastru je vhodný pouze pro systémy, kde se aplikace spotřebovávající méně často připojují nebo odpojují; například aplikace s dlouhou dobou zpracování. Při použití k monitorování systémů, kde jsou aplikace, které spotřebovávají, připojeny pouze na krátká období, může latence vzniklá při distribuci aktualizací konfigurace vést k tomu, že správci front v klastru budou mít nesprávný pohled na fronty, k nimž jsou připojeni spotřebitelé. Tato latence může mít za následek nesprávně směrované zprávy.

Při monitorování mnoha front může relativně nízká četnost změn v připojených spotřebitelích napříč všemi frontami zvýšit provoz konfigurace klastru v rámci klastru. Zvýšený provoz konfigurace klastru může vést k nadměrnému zatížení jednoho nebo více následujících správců front.

- Správci front, kde je spuštěn ukázkový program monitorování front 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.

Je třeba posoudit využití procesoru ve správcích front úplného úložiště. Další využití procesoru je viditelné jako provoz zpráv ve frontě úplného úložiště SYSTEM.CLUSTER.COMMAND.QUEUE. Pokud se v této frontě sestavují zprávy, znamená to, že správci front úplného úložiště nejsou schopni držet krok s rychlostí změny konfigurace klastru v systému.

Je-li mnoho front monitorováno ukázkovým programem pro monitorování front klastru, je zde množství práce, kterou provádí ukázkový program a správce front. Tato práce se provádí, i když neexistují žádné změny připojených spotřebitelů. Argument **-i** lze upravit tak, aby se snížilo využití procesoru ukázkového programu na lokálním systému, a to snížením frekvence cyklu monitorování.

Aby bylo možné zjistit nadměrnou aktivitu, ukázkový program monitorování fronty klastru vykazuje průměrnou dobu zpracování za interval výzev, uplynulou dobu zpracování a počet změn konfigurace. Sestavy jsou doručeny v informační zprávě, **CLM0045I**, každých 30 minut nebo každých 600 intervalů výzev, podle toho, co nastane dříve.

Požadavky na využití monitorování fronty klastru

Ukázkový program monitorování fronty klastru má požadavky a omezení. Poskytnutý ukázkový zdrojový kód můžete upravit, abyste změnili některá z těchto omezení v tom, jak je lze použít. Příklady uvedené v této části uvádějí podrobné úpravy, které lze provést.

- Ukázkový program pro monitorování fronty klastru je navržen tak, aby se používal k monitorování front, kde jsou aplikace, které spotřebovávají, buď připojeny, nebo nejsou připojeny. Pokud má systém náročné aplikace, které se často připojují a odpojují, ukázkový program může 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 základním systému IBM MQ a technologii klastru. Počet monitorovaných front, frekvence monitorování a frekvence změny stavu každé fronty

ovlivňují zátěž na celkovém systému. Tyto faktory je třeba vzít v úvahu při výběru front, které mají být monitorovány, a intervalu výzev monitorování.

- Instance ukázkového programu monitorování fronty klastru musí být připojena ke každému správci front v klastru, který vlastní instanci fronty, jež má být monitorována. Ukázkový program není nutné připojovat ke správcům front v klastru, kteří fronty nevlastní.
- Ukázkový program monitorování fronty klastru musí být spuštěn s vhodnou autorizací pro přístup ke všem požadovaným prostředkům IBM MQ . Například:
  - Správce front, ke kterému se má připojit
  - Systém SYSTEM.ADMIN.COMMAND.QUEUE
  - Všechny fronty, které mají být monitorovány při provádění přenosu zpráv
- Příkazový server musí být spuštěn pro každého správce front s připojeným ukázkovým programem pro monitorování front klastru.
- Každá instance ukázkového programu monitorování front klastru vyžaduje výhradní použití lokální (neklastrované) fronty ve správci front, ke kterému je připojena. Tato lokální fronta se používá k řízení ukázkového programu a přijímání zpráv s odpovědí z dotazů provedených na příkazovém serveru správce front.
- Všechny fronty, které mají být monitorovány jednou instancí ukázkového programu monitorování fronty klastru, musí být ve stejném klastru. Pokud má správce front fronty ve více klastrech, které vyžadují monitorování, je vyžadováno více instancí ukázkového programu. Každá instance potřebuje lokální frontu pro zprávy řízení a odpovědi.
- Všechny fronty, které se mají monitorovat, 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 front klastru. Pokud pouze podmnožina monitorovaných front vyžaduje povolený přenos zpráv, jsou potřeba dvě instance ukázkového programu monitorování front klastru. Jeden ukázkový program má povolen přenos zpráv a druhý má zakázaný přenos zpráv. Každá instance ukázkového programu potřebuje lokální frontu pro zprávy řízení a odpovědi.
- Vyrovnávání pracovní zátěže klastru IBM MQ bude standardně odesílat zprávy instancím klastrovaných front, které se nacházejí ve stejném správci front, k němuž je připojená vkládající aplikace. Toto musí být zakázáno, když je lokální fronta neaktivní za následujících okolností:
  - Vkládající aplikace se připojují ke správcům front, kteří vlastní instance neaktivní fronty, jež jsou monitorovány.
  - Zprávy ve frontě jsou přenášeny z neaktivních front do aktivních front.

  Předvolbu lokálního vyrovnávání pracovní zátěže ve frontě lze zakázat staticky, a to nastavením hodnoty CLWLUSEQ na hodnotu ANY. V této konfiguraci jsou zprávy vkládané do lokálních front distribuovány do lokálních a vzdálených instancí front za účelem vyvážení pracovní zátěže, a to i v případě, že existují lokální aplikace, které spotřebovávají. Případně lze ukázkový program monitorování fronty klastru nakonfigurovat tak, aby dočasně nastavil hodnotu **CLWLUSEQ** na ANY , když fronta nemá žádné připojené spotřebitele, což má za následek, že pouze lokální zprávy budou směřovat do lokálních instancí fronty, když je tato fronta aktivní.
- Systém IBM MQ a aplikace nesmí používat **CLWLPRTY** pro fronty, které mají být monitorovány, nebo pro používané kanály. Jinak akce ukázkového programu monitorování fronty klastru na atributech fronty **CLWLPRTY** mohou mít nežádoucí účinky.
- Ukázkový program monitorování fronty klastru protokoluje běhové informace do sady souborů sestav. Je vyžadován 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 spustit buď lokálně připojenou ke správci front, nebo jako klienta připojeného prostřednictvím kanálu. Ukázka by měla být spuštěna vždy, když je spuštěn správce front, a při lokálním spuštění může být konfigurována jako služba správce front pro automatické spuštění a zastavení ukázky se správcem front.

Než začnete

Před spuštěním ukázky monitorování fronty klastru je třeba provést následující kroky.

1. Vytvořte pracovní frontu pro každého správce front pro interní použití ukázky.

   Každá instance ukázky potřebuje lokální neklastrovou frontu pro výhradní interní použití. Můžete zvolit název fronty. Příklad používá název AMQSCLM.CONTROL.QUEUE. Například v systému Windowsmůžete 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 chybových a informačních zpráv.

   Ukázka zapisuje diagnostické zprávy do souborů sestav. Musíte zvolit adresář, do kterého se mají ukládat soubory. Například v systému Windowsmůžete vytvořit adresář pomocí následujícího příkazu:

   ```
   mkdir C:\AMQSCLM\rpts
   ```

   Soubory sestav vytvořené ukázkou mají následující konvenci pojmenování:

   ```
   QmgrName.ClusterName.RPT0n.LOG
   ```

3. (Volitelné) Definujte vzorek monitorování fronty klastru jako službu IBM MQ .

   Chcete-li monitorovat fronty, musí být ukázka vždy spuštěna. Chcete-li zajistit, aby byla ukázka monitorování front klastru vždy spuštěna, můžete definovat ukázku jako službu správce front. Definování ukázky jako služby znamená, že AMQSCLM se spustí při spuštění správce front. Následující příklad můžete použít k definování ukázky monitorování fronty klastru jako služby IBM MQ .

   ```
   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	Určuje název služby. Můžete zvolit název služby.
descr	Určuje textový popis služby.
control	Označuje, že služba se spouští a zastavuje současně se správcem front.
servtype	Označuje, že pro tohoto správce front lze v daném okamžiku spustit objekt služby serveru, což znamená pouze jednu instanci.
startcmd	Určuje umístění a název programu.
startarg	Určuje argumenty ukázky. Všimněte si použití *+ QMNAME +*. Název správce front je automaticky nahrazen.
stdout	Úplný název souboru, do kterého je přesměrován standardní výstup. Ukázka zapisuje do tohoto souboru pouze zprávy potvrzující, že ukázka byla ukončena. Ukázka to provede, protože standardní chybový soubor již byl zavřen v dřívější fázi procesu ukončení ukázky.

Definice	Popis
stderr	Úplný název souboru, do kterého je přesměrován standardní chybový výstup. Ukázka zapisuje do standardního chybového souboru všechny chybové zprávy před ukončením ukázky.

Informace o této úloze

Tato úloha vám umožňuje spustit a zastavit ukázku monitorování fronty klastru různými způsoby. Také vám umožňuje 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 s ukázkou monitorování fronty klastru, spolu s dalšími informacemi o každé z nich.

Argument	Proměnná	Další informace
-m	**QMgrName**	Správce front, který se má monitorovat.
-c	**ClusterName**	Klastr obsahující fronty, které se mají monitorovat.
-q	**QNameMask**	Fronta nebo fronty, které se mají monitorovat. Koncový systém * monitoruje všechny fronty s názvy, které se shodují s žádným nebo více koncovými znaky.
-f	**QListFile**	Úplná cesta a název souboru obsahujícího seznam názvů front nebo masek názvů front, které se mají monitorovat. Soubor musí obsahovat jeden název fronty/masku na řádek. Můžete uvést **-q** nebo **-f**, ale ne obojí.
-r	**MonitorQName**	Lokální fronta, kterou používá výhradně ukázka.
-l	**ReportDir**	Cesta k adresáři, do kterého se mají ukládat protokolované informační zprávy v sadě zalamování[9]soubory sestav.
-t		(Volitelné) Povolí přenos zpráv ve frontě z neaktivních lokálních front do aktivních front. Není-li tato volba povolena, budou do aktivních instancí fronty dynamicky směrovány pouze nové zprávy vstupující do klastru.
-u	**ActiveVal**	(Volitelné) Automaticky přepne vlastnost **CLWLUSEQ** monitorované instance fronty na hodnotu ANY , když je neaktivní, a na hodnotu **ActiveVal** , když je aktivní. **ActiveVal** může být LOCAL nebo QMGR. Není-li tento argument nastaven v systému, kde se vkládající aplikace připojují ke stejnému správci front nebo kde je povolen přenos zpráv, musí mít monitorované fronty hodnotu **CLWLUSEQ** ANYnebo QMGR , přičemž správce front má hodnotu ANY.
-i	**Interval**	(Volitelné) Časový interval v sekundách, ve kterém monitor kontroluje fronty. Výchozí hodnota je 300 sekund (5 minut).
-d		(Volitelné) Povolí další diagnostický výstup. Výstup ladění může být užitečný při počáteční konfiguraci systému nebo při práci s ukázkovým kódem.
-s		(Volitelné) Povoluje minimální statistický výstup za interval.
-v		(Volitelné) Kromě souborů sestavy protokolovat informace sestavy do souboru `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
```

Příklad souboru se seznamem front:

```
Q1
QUEUE.*
ABC
ABD
```

[9] Pro každou kombinaci správce front a front je vygenerován soubor protokolu s pevnou velikostí, který je při zaplnění přepsán. Zapisovač protokolu vždy zapisuje do stejného souboru a také uchovává 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.
   - Použijte příkaz MQSC **START SERVICE** , pokud je ukázka konfigurována jako služba IBM MQ .

   Seznam argumentů je v obou případech stejný.

   Ukázka nespustí monitorování front po dobu 10 sekund po inicializaci programu. Tato prodleva umožňuje aplikacím, které spotřebovávají, připojit se nejprve k monitorovaným frontám a zabránit tak zbytečným změnám aktivního stavu fronty.
2. Zastavte ukázku monitorování fronty klastru. Ukázka se automaticky zastaví při zastavení, zastavení, uvedení do klidového stavu nebo při přerušení připojení ke správci front. Existují způsoby, jak zastavit ukázku bez ukončení správce front:
   - Chcete-li zakázat funkci Get, nakonfigurujte lokální frontu používanou výhradně ukázkou.
   - Odešlete zprávu s hodnotou **CorrelId** `"STOP CLUSTER MONITOR\0\0\0\0"`do lokální fronty používané výhradně ukázkou.
   - Ukončete ukázkový proces. To může vést ke ztrátě dočasných zpráv přenášených do aktivních front. Může také vést k tomu, že lokální fronta použitá ukázkou bude po dobu několika sekund po ukončení zadržena jako otevřená. Tato situace brání okamžitému spuštění nové instance ukázky monitorování fronty klastru.

   Pokud byla ukázka spuštěna jako služba IBM MQ , **STOP SERVICE** nemá žádný účinek. Je možné použít jednu z metod ukončení popsaných ve správci front jako konfigurovaný mechanizmus **STOP SERVICE** .

Jak pokračovat dále

Zkontrolujte stav ukázky.

Je-li vytváření sestav povoleno, můžete zkontrolovat stav souborů sestav. K přezkoumání nejaktuálnějšího souboru sestavy použijte následující příkaz:

```
QMgrName.ClusterName.RPT01.LOG
```

Chcete-li zkontrolovat starší soubory sestav, použijte následující příkazy:

```
QMgrName.ClusterName.RPT02.LOG
QMgrName.ClusterName.RPT03.LOG
```

Soubory sestav rostou na maximální velikost přibližně 1 MB. Když se soubor RPT01 naplní, vytvoří se nový soubor RPT01 . Starý soubor RPT01 se přejmenuje na RPT02. RPT02 je přejmenován na RPT03. Starý soubor RPT03 je vyřazen.

Ukázka vytváří informační zprávy v následujících situacích:

- při spuštění
- při ukončení
- když označuje frontu **ACTIVE** nebo **INACTIVE**
- když požaduje zprávy z neaktivní fronty do aktivní instance nebo instancí

Ukázka vytvoří chybovou zprávu *CLMnnnnE* pro ohlášení problému, který vyžaduje pozornost.

Každých 30 minut ukázka vykazuje průměrnou dobu zpracování za interval výzev a uplynulou dobu zpracování. Tyto informace jsou zadrženy ve zprávě CLM0045I.

Když jsou povoleny statistické zprávy **-s**, ukázka ohlásí následující statistické informace o každé kontrole fronty:

- Doba potřebná ke zpracování front (v milisekundách)

- Počet zkontrolovaných front
- Počet provedených aktivních/neaktivních změn
- Počet přenesených zpráv

Tyto informace jsou uvedeny ve zprávě CLM0048I.

Soubory sestav mohou v režimu ladění rychle narůstat 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 rozpoznány při použití ukázky. K dispozici jsou informace o potenciálních vysvětleních pro scénář a volby, jak jej vyřešit.

Scénář: AMQSCLM se nespouští

Potenciální vysvětlení: Chybná syntaxe.

Akce: Zkontrolujte, zda standardní chybový výstup neobsahuje správnou syntaxi.

Potenciální vysvětlení: Správce front není k dispozici.

Akce: Zkontrolujte, zda soubor sestavy neobsahuje ID zprávy CLM0010E.

Potenciální vysvětlení: Nelze otevřít nebo vytvořit soubor nebo soubory sestavy.

Akce: Během inicializace zkontrolujte standardní chybový výstup, zda neobsahuje chybové zprávy.

Scénář: AMQSCLM nemění frontu na AKTIVNÍ nebo NEAKTIVNÍ

Potenciální vysvětlení: Fronta není 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 pro tohoto správce front a klastr spuštěn.

Akce: Spusťte AMQSCLM pro příslušného správce front a klastr.

Potenciální vysvětlení: Fronta je ponechána NEAKTIVNÍ, **CLWLPRTY** = 0, protože nemá žádné spotřebitele. Alternativně je ponechána AKTIVNÍ **CLWLPRTY** > =1, protože má alespoň 1 spotřebitele.

Akce: Zkontrolujte, zda jsou aplikace, které spotřebovávají, připojeny k frontě.

Potenciální vysvětlení: Příkazový server správce front není spuštěn.

Akce: Zkontrolujte, zda soubory sestav neobsahují chyby.

Scénář: Zprávy nejsou směrovány kolem NEAKTIVNÍCH front

Potenciální vysvětlení: Zprávy jsou vloženy přímo do správce front, který vlastní neaktivní frontu, a hodnota **CLWLUSEQ** fronty není ANYa argument **-u** se nepoužívá pro AMQSCLM.

Akce: Zkontrolujte hodnotu **CLWLUSEQ** příslušného správce front nebo se ujistěte, že je pro AMQSCLM použit argument **-u** .

Potenciální vysvětlení: Ve správcích front nejsou žádné aktivní fronty. Zprávy jsou rovnoměrně rozloženy v pracovní zátěži ve všech neaktivních frontách, dokud se fronta nestane aktivní.

Akce: Zkontrolujte stav front ve všech správcích front.

Potenciální vysvětlení: Zprávy jsou vloženy do jiného správce front v klastru do správce front, který je vlastníkem neaktivní fronty, a aktualizovaná hodnota **CLWLPRTY** 0 není šířena do správce front vkládající aplikace.

Akce: Zkontrolujte, zda jsou spuštěny kanály klastru mezi monitorovaným správcem front a správcem front úplného úložiště. Zkontrolujte, zda jsou spuštěny kanály mezi správcem front vkládání a správcem front úplného úložiště. Zkontrolujte protokoly chyb monitorovaných správců front, správců front vložení a správců front úplného úložiště.

Potenciální vysvětlení: Instance vzdálené fronty 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 nepřenáší zprávy z neaktivní fronty

Potenciální vysvětlení: Přenos zpráv není povolen (**-t**).

Akce: Ujistěte se, že je přenos zpráv povolen (**-t**).

Potenciální vysvětlení: Fronta není v seznamu front, které mají být monitorovány.

Akce: Zkontrolujte hodnoty parametrů **-q** a **-f** .

Potenciální vysvětlení: AMQSCLM není pro tuto nebo jiné správce front v klastru, kteří vlastní instance stejné fronty, spuštěn.

Akce: Spusťte AMQSCLM.

Potenciální vysvětlení: Fronta má hodnotu **CLWLUSEQ** = LOCAL nebo **CLWLUSEQ** = QMGRa argument **-u** není nastaven.

Akce: Nastavte parametr **-u** nebo změňte konfiguraci fronty či 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 vzdálené fronty mají spotřebitele (**IPPROCS** > = 1), ale jsou v těchto správcích front neaktivní (**CLWLPRTY** = 0), protože AMQSCLM tyto vzdálené instance nemonitoruje.

Akce: Zkontrolujte, zda je v těchto správcích front spuštěn příkaz AMQSCLM a zda je fronta v seznamu front, které mají být monitorovány, pomocí hodnot parametrů **-q** a **-f** .

Potenciální vysvětlení: Instance vzdálené fronty jsou aktivní (**CLWLPRTY** = 1), ale jsou považovány za neaktivní v lokálním správci front (**CLWLPRTY** = 0). Tato situace je způsobena tím, že aktualizovaná hodnota **CLWLPRTY** nebyla šířena do tohoto správce front.

Akce: Ověřte, zda jsou vzdálení správci front připojeni alespoň k jednomu správci front úplného 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 je nelze načíst.

Akce: Zkontrolujte, zda odesílající aplikace správně funguje.

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.

Potenciální vysvětlení: AMQSCLM zjistil chybu.

Akce: Zkontrolujte, zda soubory sestav neobsahují chyby.

Potenciální vysvětlení: Instance vzdálené fronty jsou aktivní (CLWLPRTY=1), ale zprávy nelze přenést do těchto instancí fronty, protože odesílací kanál klastru z lokálního správce front není spuštěn. Toto je často doprovázeno varováním 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.

ALW *Ukázkový program pro vyhledání koncového bodu připojení (CEPL)*

IBM MQ Ukázka vyhledání koncového bodu připojení poskytuje jednoduchý, ale výkonný modul ukončení, který uživatelům produktu IBM MQ nabízí způsob, jak načíst definice připojení z úložiště LDAP, jako např. Tivoli Directory Server.

Tivoli Directory Server v6.3 Klient musí být nainstalován, aby mohl používat CEPL.

Chcete-li použít tuto ukázku, potřebujete pracovní znalosti administrace produktu IBM MQ na podporovaných platformách.

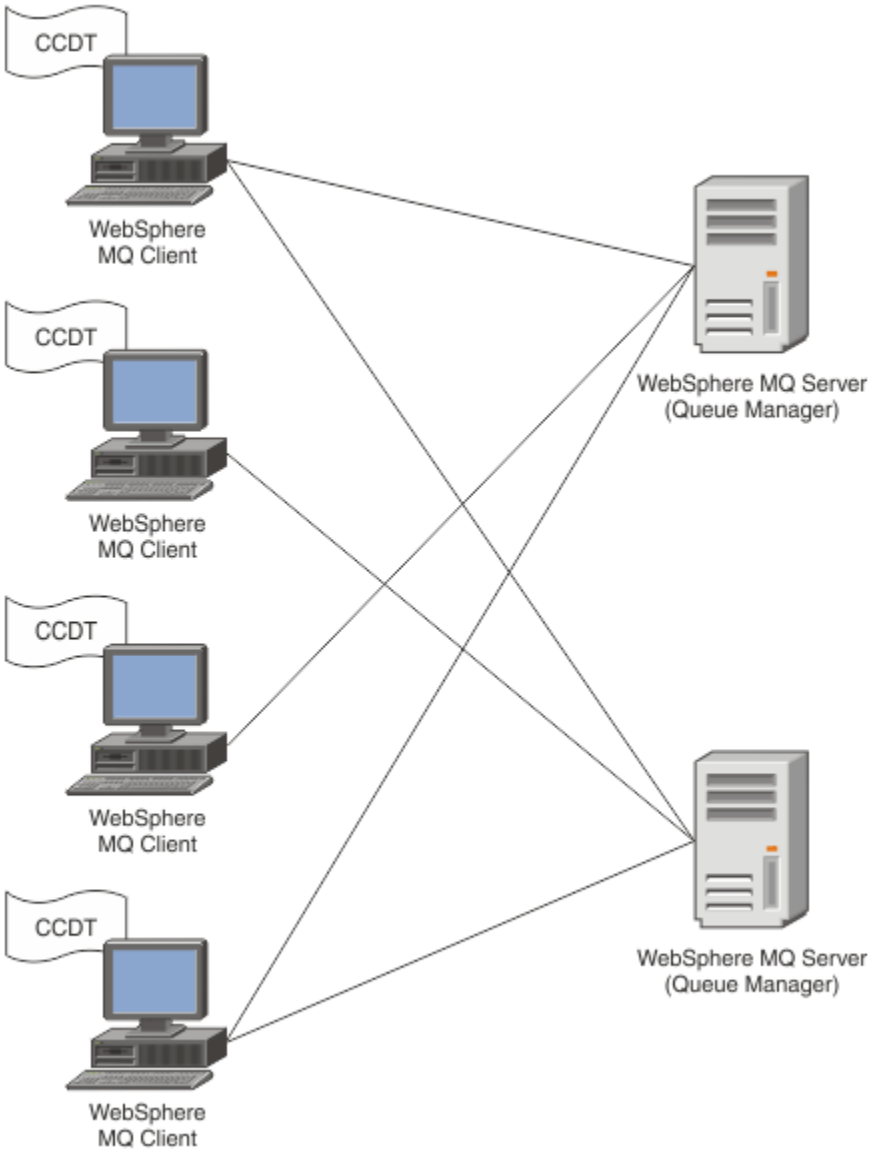
Windows Linux AIX *Úvod*

Konfigurujte globální úložiště, například adresář LDAP (Lightweight Directory Access Protocol), pro uložení definic připojení klienta pro podporu údržby a administrace.

Použití aplikace klienta IBM MQ k vytvoření připojení ke správci front prostřednictvím tabulky CCDT (Client Connection Definition Table).

Tabulka CCDT se vytváří prostřednictvím standardního rozhraní administrace produktu IBM MQ MQSC. Aby bylo možné vytvořit definice připojení klienta, musí být uživatel připojen ke správci front, i když data obsažená v definici nejsou omezena na správce front.

Vygenerovaný soubor CCDT musí být ručně distribuován mezi klientské počítače a aplikace.

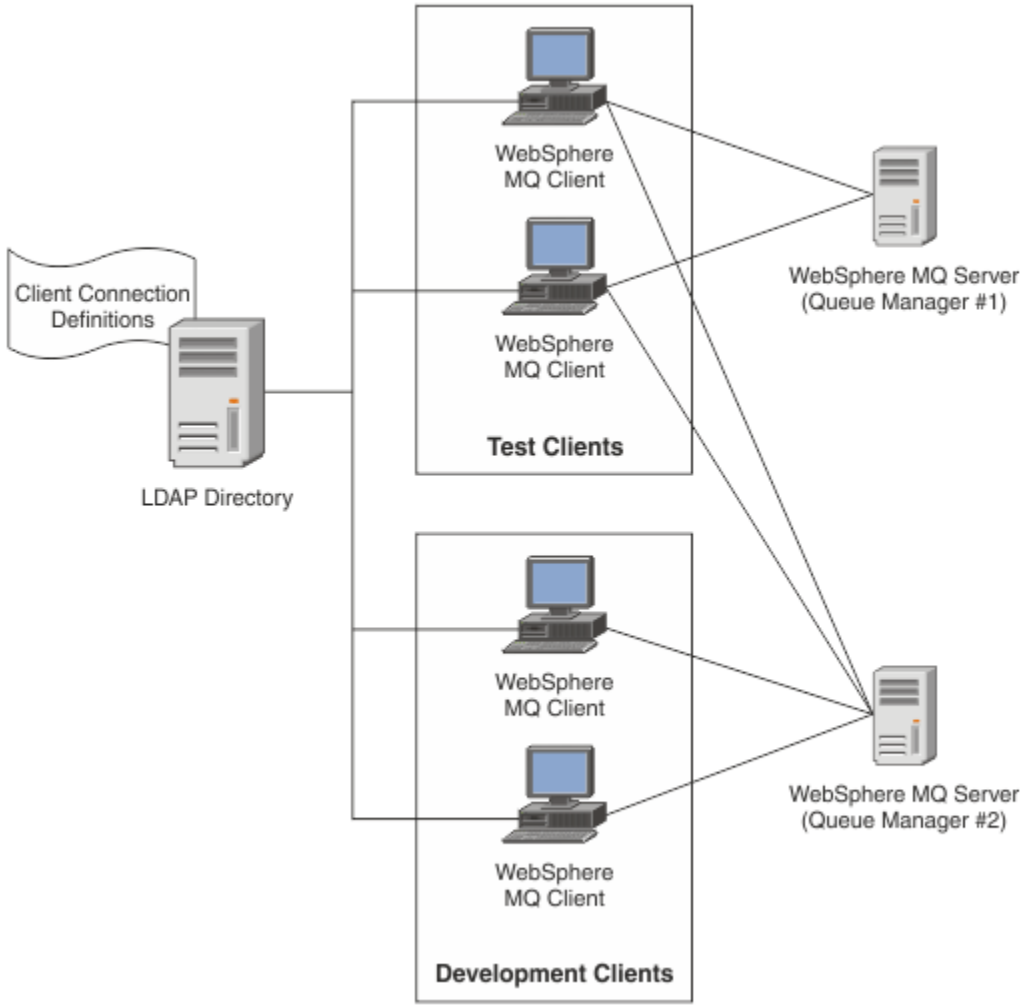


Soubor CCDT musí být distribuován na každého klienta IBM MQ . Tam, kde tisíce klientů mohou existovat buď lokálně, nebo globálně, bude brzy obtížné je udržovat a spravovat. Je zapotřebí flexibilnější přístup, který pomůže zajistit, aby měl každý klient k dispozici správné definice klienta.

Jedním z takových přístupů je uložení definic připojení klienta do globálního úložiště, například do adresáře LDAP (Lightweight Directory Access Protocol). Adresář LDAP může také poskytovat další prostředky zabezpečení, indexování a vyhledávání, což umožňuje každému klientovi přístup pouze k definicím připojení, které se jich týkají.

Adresář LDAP lze konfigurovat tak, aby určité skupiny uživatelů měly k dispozici pouze specifické definice. Testovací klienti mohou například přistupovat ke správci front #1

i k produktu #2, zatímco vývojoví klienti mohou přistupovat pouze ke správci front #2 .



Modul uživatelské procedury může vyhledat úložiště LDAP, například IBM Tivoli Directory Server, pro načtení definic kanálů. Pomocí těchto definic připojení může aplikace klienta IBM MQ vytvořit připojení ke správci front.

Modul uživatelské procedury je modul uživatelské procedury před připojením, který umožňuje získat definici kanálu během volání MQCONN/MQCONNX z úložiště LDAP.

Modul uživatelské procedury a schéma mohou být implementovány:

- Zákazníci, kteří již vybudovali dovednostní základnu s využitím stávajících technologií založených na souborech CCDT a chtějí snížit administrativní a distribuční náklady.
- Stávající zákazníci, kteří již používají vlastní technologii propriety pro distribuci definic připojení klientů.
- Noví nebo stávající zákazníci, kteří v současné době nepoužívají žádný typ řešení pro připojení klientů a chtějí využívat funkce nabízené společností IBM MQ.
- Noví nebo stávající zákazníci, kteří chtějí přímo použít nebo vyladit svůj model systému zpráv vložený s jakoukoli aktuální obchodní architekturou LDAP.

ALW *Podporovaná prostředí*

Před spuštěním ukázky Vyhledávání koncového bodu připojení ověřte, že máte podporovaný operační systém a příslušný software.

Ukázkový program pro IBM MQ Connection Endpoint Lookup vyžaduje následující software:

- IBM WebSphere MQ 7.0nebo novější

- Tivoli Directory Server V6.3 Client nebo novější

Podporované operační systémy:

1. 

Windows (7. 8. 2008/2012)
2. 

AIX
3. 

Linux
   - RHEL v4 a v5 na systému System p
   - SUSE v9 a v10 na systému System p
   - RHEL v4 a v5 x86-64 32bitová a 64bitová verze
   - SUSE v9 a v10 x86-64 32bitová a 64bitová

Poznámka: Ukázka není k dispozici pro následující platformy:

- 

z/OS
- 

IBM i

Instalace a konfigurace
Instalace a konfigurace modulu uživatelské procedury a schématu koncového bodu připojení.

Instalace modulu uživatelské procedury

Během instalace produktu IBM MQje modul uživatelské procedury nainstalován v adresáři `tools/samples/c/preconnexit/bin`. Pro 32bitové platformy musí být před použitím zkopírován modul ukončení do adresáře `exit/`*`installation_name`*`/` . Pro 64bitové platformy musí být modul uživatelské procedury před použitím zkopírován do adresáře exit64/*installation_name*/.

Instalace schématu koncového bodu připojení

Uživatelská procedura používá schéma koncového bodu připojení `ibm-amq.schema`. Před použitím uživatelské procedury musí být soubor schématu importován na libovolný server LDAP. Po importu schématu musí být přidány hodnoty pro atributy.

Zde je příklad importu schématu koncového bodu připojení. Příklad předpokládá, že se používá produkt IBM Tivoli Directory Server (ITDS).

- Ujistěte se, že je spuštěn server IBM Tivoli Directory Server , pak zkopírujte nebo FTP soubor `ibm-amq.schema` na server ITDS.
- Na serveru ITDS zadejte následující příkaz pro instalaci schématu do úložiště ITDS, kde *ID LDAP* 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 naleznete 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 obsahujícího kód uživatelské procedury 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 kód ukončení `LdapPreConnect` . Definice funkce se řídí prototypem funkce vašeho podniku.

Upozornění: Měli byste odstranit uvozovky v příkazu funkce, když uvádíte skutečný vstupní bod ukončení.

Data
Identifikátor URI úložiště LDAP obsahujícího definice kanálu.

Následující úsek kódu je příkladem změn požadovaných v souboru `mqclient.ini` .

```
PreConnect:
Module=amqlcelp
Function="LdapPreconnectExit"
Data=ldap:dap://myLDAPServer.com:389/cn=wmq,ou=ibm,ou=com
Sequence=1
```

ALW *Přehled uživatelské procedury a schématu*
Syntaxe a parametry použité k vytvoření připojení ke správci front.

IBM MQ 9.3 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 produkt IBM MQ C Client načte modul uživatelské procedury obsahující implementaci syntaxe funkce. Poté vyvolá funkci ukončení pro načtení definic kanálů. Načtené definice kanálů jsou poté použity k vytvoření připojení ke správci front.

Parametry

pExitParms (parametry)
Typ: Vstup/výstup PMQNXP

Struktura parametrů ukončení PreConnection . Struktura je přidělena a udržována volajícím na výstupu.

```
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: Vstupní/výstup typu PMQCHAR

Název správce front. Na vstupu je tento parametr řetězec filtru dodaný do volání rozhraní API MQCONN prostřednictvím parametru **QMgrName** . Toto pole může být prázdné, explicitní nebo může obsahovat určité zástupné znaky. Pole se změní uživatelskou procedurou. Parametr je NULL , když je uživatelská procedura volána s MQXR_TERM.

ppConnectvolby

Typ: ppConnectVstupní/výstupní volby

Volby, které řídí akci MQCONNX. Jedná se o ukazatel na strukturu voleb připojení MQCNO, která řídí akci volání rozhraní API MQCONN. Parametr je NULL , když je uživatelská procedura volána s MQXR_TERM. Klient MQI vždy poskytuje uživatelskou proceduru strukturu MQCNO, a to i v případě, že nebyla původně poskytnuta aplikací. Pokud aplikace poskytuje strukturu MQCNO, klient vytvoří duplikát, aby ji předal uživatelské proceduře, kde je upravena. Klient si ponechává vlastnictví MQCNO. MQCD odkazované prostřednictvím MQCNO má přednost před jakoukoli definicí připojení poskytnutou prostřednictvím pole. Klient používá strukturu MQCNO pro připojení ke správci front a ostatní jsou ignorovány.

Kód pComp

Typ: Vstupní/výstupní PMQLONG

Kód dokončení. Ukazatel na objekt MQLONG, který přijímá kód dokončení ukončení. Musí to být jedna z následujících hodnot:

- MQCC_OK -Úspěšné dokončení
- MQCC_WARNING -Varování (částečné dokončení)
- MQCC_FAILED -Volání se nezdařilo.

pReason

Typ: Vstupní/výstupní PMQLONG

Kód příčiny pComp. Ukazatel na MQLONG, který obdrží kód příčiny ukončení. Pokud je kód dokončení MQCC_OK, jediná platná hodnota je: MQRC_NONE -(0, x '000') Není důvod k hlášení.

Pokud je kód dokončení MQCC_FAILED nebo MQCC_WARNING, funkce ukončení může nastavit pole kódu příčiny na libovolnou platnou hodnotu MQRC_ *.

ALW *MQ Informace o kontextu LDAP*

Uživatelská procedura používá pro informace o kontextu následující datovou strukturu.

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 Linux AIX *Vzorový kód pro sestavení uživatelské procedury pro vyhledání koncového bodu připojení*

Můžete použít úseky kódu ukázky pro kompilaci zdroje v systému AIX, Linuxnebo Windows.

Kompilace zdroje

Zdroj můžete kompilovat s libovolnými knihovnami klienta LDAP, například s knihovnami klienta IBM Tivoli Directory Server V6.3 . Tato dokumentace předpokládá, že používáte knihovny klienta produktu Tivoli Directory Server V6.3 .

Poznámka: Knihovna uživatelské procedury před připojením je podporována s následujícími servery LDAP:

- IBM Tivoli Directory Server V6.3
- Novell eDirectory V8.2

Následující úseky kódu popisují, jak kompilovat uživatelské procedury:

Windows **Kompilace uživatelské procedury na platformě Windows**

Pro kompilaci zdroje uživatelské procedury 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: Používáte-li IBM Tivoli Directory Server V6.3 knihovny klienta, které jsou kompilovány pomocí kompilátoru Microsoft Visual Studio 2003 , můžete při kompilaci produktu IBM Tivoli Directory Server V6.3 Client libraries with Microsoft Visual Studio 2012nebo novější, zobrazit varování.

Linux AIX **Kompilace uživatelské procedury v systému AIX, Linux**

Následující úsek kódu je určen pro kompilaci zdroje uživatelské procedury v systému Linux. Některé volby kompilátoru se mohou v produktu AIXlišit.

```
#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 dodává knihovny statických i dynamických odkazů, 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)
```

ALW *Vyvolání modulu ukončení PreConnect*

Modul uživatelské procedury PreConnect lze vyvolat se třemi různými kódy příčiny: kódem příčiny MQXR_INIT pro inicializaci a vytvoření připojení k serveru LDAP, kódem příčiny MQXR_PRECONNECT pro načtení definic kanálů ze serveru LDAP nebo kódem příčiny MQXR_TERM při vyčištění uživatelské procedury.

MQXR_INIT

Uživatelská procedura je vyvolána s kódem příčiny MQXR_INIT pro inicializaci a vytvoření připojení k serveru LDAP.

Před voláním MQXR_INIT je pole pExitDataPtr struktury MQNXP naplněno atributem Data ze sekce PreConnect v souboru `mqclient.ini` (tj. LDAP).

URL LDAP se skládá alespoň z protokolu, názvu hostitele, čísla portu a základního rozlišujícího názvu pro hledání. Uživatelská procedura analyzuje URL LDAP obsaženou v poli pExitDataPtr , přidělí strukturu kontextu vyhledávání LDAP MQNLDAPCTX a naplní ji odpovídajícím způsobem. Adresa této struktury je uložena v poli Ptr pExitUserArea. Selhání správné analýzy adresy URL LDAP URL má za následek chybu MQCC_FAILED.

V tomto bodě se uživatelská procedura připojí a připojí k serveru LDAP pomocí parametrů **MQNLDAPCTX** . Výsledné popisovače rozhraní API LDAP jsou také uloženy v této struktuře.

MQXR_PRECONNECT

Modul uživatelské procedury je vyvolán s kódem příčiny MQXR_PRECONNECT pro načtení definic kanálů ze serveru LDAP.

Uživatelská procedura vyhledává na serveru LDAP definice kanálů odpovídající danému filtru. Pokud **QMgrNameparameter** obsahuje specifický název správce front, hledání vrátí všechny definice kanálů, pro které hodnota atributu LDAP **ibm-amqQueueManagerName** odpovídá danému názvu správce front.

Je-li parametr **QMgrName** '*' nebo ' ' (prázdný), pak hledání vrátí všechny definice kanálů, pro které je atribut koncového bodu **ibm-amqIsClientDefault Connection** nastaven na hodnotu TRUE.

Po úspěšném hledání uživatelská procedura připraví jednu nebo pole definic MQCD a vrátí se zpět volajícímu.

MQXR_TERM

Uživatelská procedura je vyvolána s tímto kódem příčiny, když má být uživatelská procedura vyčištěna. Během tohoto čištění se uživatelská procedura odpojí od serveru LDAP a uvolní veškerou paměť přidělenou a udržovanou uživatelskou procedurou, včetně struktury MQNLDAPCTX, pole ukazatele a každého disku MQCD, na který odkazuje. Všechna ostatní pole jsou nastavena na výchozí hodnoty. Parametry uživatelské procedury **pQMgrName** a **ppConnectOpts** se během uživatelské procedury nepoužívají s kódem příčiny MQXR_TERM a mohou mít hodnotu NULL.

Související odkazy

Sekce PreConnect konfiguračního souboru klienta

ALW *Schémata LDAP*

Data připojení klienta jsou uložena v globálním úložišti nazvaném adresář LDAP (Lightweight Directory Access Protocol). Klient IBM MQ používá adresář LDAP k získání definic připojení. Struktura definic připojení klienta IBM MQ v adresáři LDAP je známa jako schéma LDAP. Schéma LDAP je kolekce definic typů atributů, definic tříd objektů a dalších informací, které server používá k určení, zda se deklarace hodnoty filtru nebo atributu shoduje s atributy položky, a zda povolit, přidat a upravit operace.

Ukládání dat do adresáře LDAP

Definice připojení klienta jsou umístěny pod specifickou větví v rámci adresářového stromu známého jako přípojný bod. Stejně jako všechny ostatní uzly v adresáři LDAP má přípojný bod k sobě přidružen rozlišující název (DN). Tento uzel můžete použít jako výchozí bod pro všechny dotazy, které provedete v adresáři. Filtrování použijte při dotazování adresáře LDAP, aby se vrátila podmnožina definic připojení klienta. Přístup k podstromům můžete omezit na základě oprávnění udělených v jiných částech adresářového stromu-například uživatelům, oddělením nebo skupinám.

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 číslem identifikátoru objektu (OID), které jedinečně identifikuje objekt nebo atribut. Všechny třídy ve schématu LDAP dědí buď přímo, nebo nepřímo od horního objektu. Objekt definice kanálu klienta obsahuje atributy objektu nejvyšší úrovně. 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 jsou popsány v části Atributy LDAP.

ALW *Atributy LDAP*

Definované atributy LDAP jsou specifické pro produkt IBM MQ a mapují se přímo na vlastnosti připojení klienta.

IBM MQ Atributy řetězce adresáře kanálu klienta

Atributy znakového řetězce s jejich mapováním na vlastnosti IBM MQ jsou uvedeny v následující tabulce. Atributy mohou obsahovat hodnoty syntaxe directoryString (kódování Unicode v kódováníUTF-8 , tj. systém kódování bajtů proměnných, který obsahuje jako podmnožinu IA5/ASCII). Syntaxe je určena identifikačním číslem objektu (OID).

Tabulka 166. IBM MQ atributy řetězce adresáře kanálu klienta

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-amqLocalAdresa	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 aplikace klienta IBM MQ požadovat připojení.	QMNAME
ibm-amqSecurityExitUserData	Uživatelská data, která jsou předána uživatelské proceduře pro zabezpečení zprávy.	SCYDATA
ibm-amqSecurityExitName	Název uživatelského programu, který má být spuštěn uživatelskou procedurou pro zabezpečení zprávy kanálu.	SCYEXIT
ibm-amqSslCipherSpec	Jedna CipherSpec pro připojení TLS.	SSLCIPH
ibm-amqSslPeerName	Zkontroluje rozlišující název (DN) certifikátu od správce front typu peer nebo klienta na druhém konci kanálu IBM MQ .	SSLPEER
ibm-amqTransactionProgramName	Název transakčního programu.	TPNAME

Tabulka 166. IBM MQ atributy řetězce adresáře kanálu klienta (pokračování)		
atribut LDAP	**Popis**	**IBM MQ Vlastnost**
ibm-amqUserID	ID uživatele, které má agent MCA použít při pokusu o zahájení zabezpečené relace SNA se vzdáleným agentem MCA.	USERID

IBM MQ celočíselné atributy připojení klienta

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 konstantního názvu.

Tabulka 167. Celočíselné atributy adresáře kanálu klienta IBM MQ		
atribut LDAP	**Popis**	**IBM MQ Vlastnost**
ibm-amqConnectionafinita	Určuje, zda klientské aplikace, které se připojují vícekrát prostřednictvím stejného názvu správce front, používají stejný kanál klienta.	AFFINITY
ibm-amqClientChannelWeight	Váha ovlivňující použití definice 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 přenosu, který se má použít.	TRPTYPE

IBM MQ logický atribut kanálu klienta

Tento logický atribut není mapován na žádnou vlastnost IBM MQ . Syntaxe tohoto atributu označuje logickou hodnotu.

Tabulka 168. IBM MQ logický atribut kanálu klienta	
atribut LDAP	**Popis**
ibm-amqIsClientDefault	Tento logický atribut je definován tak, aby vyřešil problém s hledáním položek, jejichž atribut ibm-amqQueueManagerName nebyl definován.

IBM MQ atributy seznamu kanálů klienta

Vlastnosti IBM MQ jsou uloženy jako jednohodnotový atribut seznamu oddělený čárkami v adresáři LDAP. Atributy jsou definovány stejným způsobem jako ostatní atributy řetězce adresáře. Atributy seznamu spolu s jejich mapováním na vlastnosti IBM MQ jsou popsány v následující tabulce.

Tabulka 169. IBM MQ atributy seznamu kanálů klienta		
atribut LDAP	**Popis**	**IBM MQ Vlastnost**
ibm-amqHeaderKomprese	Seznam technik komprese dat záhlaví podporovaných kanálem.	COMPHDR
ibm-amqMessageKomprese	Seznam technik komprese dat zpráv podporovaných kanálem.	COMPMSG
ibm-amqSendExitUserData	Uživatelská data, která jsou předána uživatelské proceduře pro odeslání.	SENDDATA
ibm-amqSendExitUser	Název uživatelského programu, který má být spuštěn uživatelskou procedurou pro odesílání kanálu.	SENDEXIT
ibm-amqReceiveExitUserData	Uživatelská data, která jsou předána uživatelské proceduře pro příjem.	RCVDATA
ibm-amqReceiveExitName	Název uživatelského programu, který má být spuštěn uživatelskou procedurou pro příjem kanálu.	RCVEXIT

ALW *Obecný název*
Obecný název (CN) se skládá z názvu kanálu a názvu definujícího správce front.

Jedná se o již existující 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 řetězcovým atributem a hodnoty nerozlišují velikost písmen. Porovnávání podřetězců je ignorováno. Porovnávání podřetězců je porovnávací pravidlo použité v podschématu, které určuje chování atributu ve filtru vyhledávání pomocí podřetězce (například CN=jim *, kde CN je atribut) a obsahuje jeden nebo více zástupných znaků.

ALW *ibm-názevamqChannel*
Tento atribut určuje název definice kanálu.

Tento atribut má jednu hodnotu řetězce s maximálně 20 znaky, které nerozlišují malá a velká písmena. Nejedná se o již existující atribut.

Porovnávání podřetězců je ignorováno. Porovnávání podřetězců je porovnávací pravidlo použité v podschématu, které určuje chování atributu ve filtru vyhledávání pomocí podřetězce a obsahuje jeden nebo více zástupných znaků.

ALW *ibm-amqDescription*
Tento atribut LDAP poskytuje popis kanálu.

Tento atribut má jednu hodnotu řetězce s maximálně 64 bajty, které nerozlišují malá a velká písmena. Nejedná se o již existující atribut.

Porovnávání podřetězců je ignorováno. Porovnávání podřetězců je porovnávací pravidlo použité v podschématu, které určuje chování atributu ve filtru vyhledávání.

ALW *ibm-amqConnectionnázev*
Tento atribut LDAP je identifikátor komunikačního připojení. Určuje konkrétní komunikační linky, které má tento kanál používat.

Tento atribut má hodnotu jednoho řetězce s maximálně 264 znaky, které nerozlišují velká a malá písmena. Nejedná se o již existující atribut.

Porovnávání podřetězců je ignorováno. Porovnávání podřetězců je porovnávací pravidlo použité v podschématu, které určuje chování atributu ve filtru vyhledávání.

ALW *ibm-amqLocalAdresa*
Tento atribut určuje lokální komunikační adresu kanálu.

Tento atribut má hodnotu jednoho řetězce s maximálně 48 znaky, které nerozlišují malá a velká písmena. Nejedná se o již existující atribut.

Porovnávání podřetězců je ignorováno. Porovnávání podřetězců je porovnávací pravidlo použité v podschématu, které určuje chování atributu ve filtru vyhledávání.

ALW *ibm-amqModeNázev*
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 hodnotu jednoho řetězce přesně 8 znaků, které nerozlišují malá a velká písmena. Nejedná se o již existující atribut.

Porovnávání podřetězců je ignorováno. Porovnávání podřetězců je porovnávací pravidlo použité v podschématu, které určuje chování atributu ve filtru vyhledávání.

ALW *ibm-amqPassword*
Tento atribut LDAP určuje heslo, které může agent MCA použít při pokusu o zahájení zabezpečené relace LU 6.2 se vzdáleným agentem MCA.

Tento atribut má jedinou celočíselnou hodnotu s maximálně 12 číslicemi. Nejedná se o již existující atribut.

ALW *ibm-amqQueueManagerName*
Tento atribut určuje název správce front nebo skupiny správců front, ke kterému může aplikace klienta IBM MQ požadovat připojení.

Tento atribut má hodnotu jednoho řetězce s maximálně 48 znaky, které nerozlišují malá a velká písmena. Nejedná se o již existující atribut.

Porovnávání podřetězců je ignorováno. Porovnávání podřetězců je porovnávací pravidlo použité v podschématu, které určuje chování atributu ve filtru vyhledávání.

Související odkazy
“ibm-amqIsClientDefault” na stránce 1120
Tento logický atribut řeší problém s hledáním položek, kde nebyl definován atribut ibm-amqQueueManagerName .

ALW *ibm-amqSecurityExitUserData*
Tento atribut LDAP určuje uživatelská data, která jsou předána uživatelské proceduře zabezpečení.

Tento atribut má jednu hodnotu řetězce s maximálně 999 znaky, které nerozlišují velká a malá písmena. Nejedná se o již existující atribut.

Porovnávání podřetězců je ignorováno. Porovnávání podřetězců je porovnávací pravidlo použité v podschématu, které určuje chování atributu ve filtru vyhledávání.

ALW *ibm-amqSecurityExitName*
Tento atribut LDAP určuje název uživatelského programu, který má být spuštěn uživatelskou procedurou pro zabezpečení zprávy kanálu.

Ponechte prázdné, pokud není v platnosti žádná uživatelská procedura zabezpečení kanálu.

Tento atribut má jednu hodnotu řetězce s maximálně 999 znaky, které nerozlišují velká a malá písmena. Tento atribut není předběžným ukončením.

Porovnávání podřetězců je ignorováno. Porovnávání podřetězců je porovnávací pravidlo použité v podschématu, které určuje chování atributu ve filtru vyhledávání.

ALW *ibm-amqSslCipherSpec*
Tento atribut LDAP určuje jedinou specifikaci CipherSpec pro připojení TLS.

Tento atribut má hodnotu jednoho řetězce s maximálně 32 znaky, které nerozlišují malá a velká písmena. Nejedná se o již existující atribut.

Porovnávání podřetězců je ignorováno. Porovnávání podřetězců je porovnávací pravidlo použité v podschématu, které určuje chování atributu ve filtru vyhledávání.

ALW *ibm-amqSslPeerName*
Tento atribut LDAP se používá ke kontrole rozlišujícího názvu (DN) certifikátu ze správce front typu peer nebo klienta na druhém konci kanálu IBM MQ .

Tento atribut LDAP má hodnotu jednoho řetězce s maximálně 1024 bajty, které nerozlišují malá a velká písmena. Nejedná se o již existující.

Porovnávání podřetězců je ignorováno. Porovnávání podřetězců je porovnávací pravidlo použité v podschématu, které určuje chování atributu ve filtru vyhledávání.

ALW *ibm-amqTransactionProgramName*
Tento atribut LDAP určuje název transakčního programu. Je určen pro použití s připojeními LU 6.2 .

Tento atribut má hodnotu jednoho řetězce s maximálně 64 znaky, které nerozlišují malá a velká písmena. Nejedná se o již existující.

Porovnávání podřetězců je ignorováno. Porovnávání podřetězců je porovnávací pravidlo použité v podschématu, které určuje chování atributu ve filtru vyhledávání.

ALW *ibm-amqUserID*
Tento atribut LDAP určuje ID uživatele, které má agent MCA použít při pokusu o zahájení zabezpečené relace SNA se vzdáleným agentem MCA.

Tento atribut má hodnotu jednoho řetězce přesně 12 znaků, které nerozlišují malá a velká písmena. Nejedná se o již existující atribut.

Porovnávání podřetězců je ignorováno. Porovnávání podřetězců je porovnávací pravidlo použité v podschématu, které určuje chování atributu ve filtru vyhledávání.

ALW *ibm-amqConnectionafinita*
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. Nejedná se o již existující atribut.

ALW *ibm-amqClientChannelWeight*
Tento atribut LDAP určuje váhu, která ovlivňuje, která definice kanálu připojení klienta se používá.

Atribut váhy kanálu klienta se používá k zkreslení výběru definic kanálů klienta, pokud je k dispozici více než jedna vhodná definice.

Tento atribut má jedinou celočíselnou hodnotu. Nejedná se o již existující atribut.

ALW *ibm-amqHeartBeatInterval*
Tento atribut LDAP určuje přibližnou dobu mezi toky synchronizačních signálů, které mají být předány z odesílajícího agenta MCA v případě, že v přenosové frontě nejsou žádné zprávy.

Tento atribut má jedinou celočíselnou hodnotu. Nejedná se o již existující atribut. Výchozí hodnota je 1. Výchozí hodnota je nastavena v aktuální operaci proměnné prostředí MQSERVER.

ALW *ibm-amqKeepAliveInterval*
Tento atribut LDAP se používá k určení hodnoty časového limitu pro kanál.

Hodnota tohoto atributu je předána do komunikačního zásobníku s uvedením časování udržení aktivity pro kanál. Pomocí této volby můžete určit jinou hodnotu udržení aktivity pro každý kanál.

Tento atribut má jedinou celočíselnou hodnotu. Nejedná se o již existující atribut.

ALW *ibm-amqMaximumMessageLength*
Tento atribut LDAP určuje maximální délku zprávy, kterou lze přenést kanálem.

Výchozí hodnota tohoto atributu je 104857600 podle aktuální operace proměnné prostředí MQSERVER. Tento atribut má jedinou celočíselnou hodnotu a není to již existující atribut.

ALW *ibm-amqSharingKonverzace*
Tento atribut LDAP určuje maximální počet konverzací, které sdílejí každou instanci kanálu TCP/IP.

Tento atribut má jedinou celočíselnou hodnotu. Tento atribut není již existujícím atributem.

ALW *ibm-amqTransportTyp*
Tento atribut LDAP uvádí typ přenosu, který se má použít.

Tento atribut má jedinou celočíselnou hodnotu. Nejedná se o již existující atribut.

ALW *ibm-amqIsClientDefault*
Tento logický atribut řeší problém s hledáním položek, kde nebyl definován atribut ibm-amqQueueManagerName .

Moduly ukončení předběžného připojení obecně vyhledávají servery LDAP s hodnotou atributu ibm-amqQueueManagerName jako vyhledávací kritéria. Takový dotaz by vrátil všechny položky, kde hodnota atributu ibm-amqQueueManagerName odpovídá názvu správce front uvedenému ve volání MQCONN/X. Při použití tabulek CCDT (Client Channel Definition Tables) však můžete buď nastavit název správce front pro volání MQCONN/X jako prázdný, nebo před název zadat hvězdičku (*). Pokud je název správce front prázdný, klient se připojí k výchozímu správci front. Má-li název ke správci front předponu hvězdička (*), klient připojí všechny správce front.

Podobně lze atribut ibm-amqQueueManagerName v položce ponechat nedefinovaný. V tomto případě se očekává, že se klient používající tyto informace o koncovém bodu může připojit k libovolnému správci front. Například položka obsahuje následující řádky:

```
ibm-amqChannelName = "CHANNEL1"
ibm-amqConnectionName = myhost(1414)
```

V tomto příkladu se klient pokusí připojit k určenému správci front spuštěnému v systému `myhost`.

Avšak na serverech LDAP není provedeno hledání hodnoty atributu, která nebyla definována. Pokud například položka obsahuje informace o připojení s výjimkou položky ibm-amqQueueManagerName, nebudou výsledky hledání tuto položku obsahovat. Chcete-li tento problém překonat, můžete nastavit ibm-amqIsClientDefault. Jedná se o logický atribut a předpokládá se, že má hodnotu FALSE, pokud není definována.

Pro položky, u kterých nebyla definována položka ibm-amqQueueManagerName a u kterých se očekává, že budou součástí vyhledávání, nastavte parametr ibm-amqIsClientDefault na hodnotu TRUE. Je-li jako název správce front ve volání MQCONN/X zadána prázdná hodnota nebo hvězdička (*), uživatelská procedura předběžného připojení vyhledá v serveru LDAP všechny položky, pro které je hodnota atributu ibm-amqIsClientDefault nastavena na hodnotu TRUE.

Poznámka: Nenastavujte ani nedefinujte atribut ibm-amqQueueManagerName , pokud je atribut ibm-amqIsClientDefault nastaven na hodnotu TRUE.

Související odkazy

“ibm-amqQueueManagerName” na stránce 1118
Tento atribut určuje název správce front nebo skupiny správců front, ke kterému může aplikace klienta IBM MQ požadovat připojení.

ALW *ibm-amqHeaderKomprese*
Tento atribut LDAP je seznam technik komprese dat záhlaví podporovaných kanálem.

Maximální velikost tohoto atributu je 48 znaků. Nejedná se o již existující atribut.

Pro tento atribut můžete zadat pouze jednu hodnotu.

Tento atribut seznamu je určen jako řetězce adresáře s použitím formátu odděleného čárkami. Například hodnota uvedená pro **ibm-amqHeaderCompression** je `0` , která je mapována na `NONE`. Všechny hodnoty, které překračují maximální povolený limit, jsou klientem ignorovány. Například komprese ibm-amqHeaderobsahuje v seznamu maximálně 2 celá čísla.

ALW *ibm-amqMessageKomprese*
Tento atribut LDAP je seznam technik komprese dat zpráv podporovaných kanálem.

Maximální velikost tohoto atributu je 48 znaků. Nejedná se o již existující atribut.

Tento atribut nepodporuje více hodnot.

Tento atribut seznamu je určen jako řetězce adresáře s použitím formátu odděleného čárkami. Například hodnota uvedená pro tento atribut je 1,2,4, která se mapuje na základní posloupnost komprese RLE, ZLIBFAST a ZLIBHIGH.

Všechny hodnoty, které překračují maximální povolený limit, jsou klientem ignorovány. Například komprese ibm-amqMessageobsahuje v seznamu maximálně 16 celých čísel.

ALW *ibm-amqSendExitUserData*
Tento atribut LDAP uvádí uživatelská data, která jsou předána uživatelské proceduře pro odeslání.

Tento atribut LDAP má jednu hodnotu řetězce s maximálně 999 znaky, které nerozlišují malá a velká písmena. Nejedná se o již existující atribut.

Porovnávání podřetězců je ignorováno. Porovnávání podřetězců je porovnávací pravidlo použité v podschématu, které určuje chování atributu ve filtru vyhledávání.

Poznámka: ibm-amqSendExitName a **ibm-amqSendExitUserData** musí být synchronizovány ve dvojicích. Uživatelská data by měla být synchronizována s názvem uživatelské procedury. Takže pokud je uveden jeden, druhý musí být také symetricky specifikován, i když neobsahuje žádná data.

ALW *ibm-amqSendExitName*
Tento atribut LDAP určuje název uživatelského programu, který má být spuštěn uživatelskou procedurou pro odesílání kanálu.

Tento atribut má jednu hodnotu řetězce s maximálně 999 znaky, které nerozlišují velká a malá písmena. Nejedná se o již existující atribut.

Porovnávání podřetězců je ignorováno. Porovnávání podřetězců je porovnávací pravidlo použité v podschématu, které určuje chování atributu ve filtru vyhledávání.

Poznámka: **ibm-amqSendExitName** a **ibm-amqSendExitUserData** musí být synchronizovány ve dvojicích. Uživatelská data musí být synchronizována s názvem uživatelské procedury. Je-li tedy uveden jeden, druhý také musí být symetricky uveden, i když neobsahuje žádná data.

ALW *ibm-amqReceiveExitUserData*
Tento atribut LDAP určuje uživatelská data, která jsou předána uživatelské proceduře pro příjem.

Můžete spustit posloupnost uživatelských procedur příjmu. Řetězec uživatelských dat pro řadu uživatelských procedur je oddělen čárkou, mezerami nebo obojím.

Tento atribut má jednu hodnotu řetězce s maximálně 999 znaky, které nerozlišují velká a malá písmena. Nejedná se o již existující atribut.

Porovnávání podřetězců je ignorováno. Porovnávání podřetězců je porovnávací pravidlo použité v podschématu, které určuje chování atributu ve filtru vyhledávání.

Poznámka: **ibm-amqReceiveExitName** a **ibm-amqReceiveExitUserData** musí být synchronizovány ve dvojicích. Uživatelská data musí být synchronizována s názvem uživatelské procedury. Je-li tedy uveden jeden, druhý také musí být symetricky uveden, i když neobsahuje žádná data.

ALW *ibm-amqReceiveExitName*
Tento atribut LDAP určuje název programu uživatelské procedury, který má být spuštěn uživatelskou procedurou pro příjem kanálu.

Tento atribut je seznam názvů programů, které mají být spuštěny v posloupnosti. Ponechte prázdné, pokud není v platnosti žádná uživatelská procedura pro příjem kanálu.

Tento atribut má jednu hodnotu řetězce s maximálně 999 znaky, které nerozlišují velká a malá písmena. Nejedná se o již existující atribut.

Porovnávání podřetězců je ignorováno. Porovnávání podřetězců je porovnávací pravidlo použité v podschématu, které určuje chování atributu ve filtru vyhledávání.

Poznámka: **ibm-amqReceiveExitName** a **ibm-amqReceiveExitUserData** musí být synchronizovány ve dvojicích. Uživatelská data musí být synchronizována s názvem uživatelské procedury. Takže pokud je uveden jeden, druhý musí být také symetricky uveden, i když neobsahuje žádná data.

z/OS Použití ukázkových programů pro z/OS

Ukázkové procedurální aplikace dodávané s produktem IBM MQ for z/OS demonstrují typické použití rozhraní MQI (Message Queue Interface).

Informace o této úloze

Produkt IBM MQ for z/OS také poskytuje ukázkové uživatelské procedury pro převod dat, které jsou popsány v tématu “Zápis uživatelských procedur převodu dat” na stránce 950.

Všechny ukázkové aplikace jsou dodávány ve zdrojové podobě; některé jsou také dodávány ve spustitelné formě. Zdrojové moduly obsahují pseudokód, který popisuje logiku programu.

Poznámka: Ačkoli některé ukázkové aplikace mají základní rozhraní řízená panely, neusilují o to, aby demonstrovaly, jak navrhnout vzhled a chování vašich aplikací. Další informace o návrhu rozhraní řízených panely pro neprogramovatelné terminály naleznete v příručce *SAA Common User Access: Basic Interface Design Guide* (SC26-4583) and its addendum (GG22-9508). Tyto pokyny vám pomohou navrhnout aplikace, které jsou konzistentní jak v rámci aplikace, tak v rámci jiných aplikací.

Procedura

- Chcete-li se dozvědět více o ukázkových programech, použijte následující odkazy:
  - “Funkce demonstrované v ukázkových aplikacích pro z/OS” na stránce 1123
  - “Příprava a spuštění ukázkových aplikací pro dávkové prostředí v systému z/OS” na stránce 1129

- “Příprava ukázkových aplikací pro prostředí TSO na systému z/OS” na stránce 1132
- “Příprava ukázkových aplikací pro prostředí CICS na systému z/OS” na stránce 1134
- “Příprava ukázkové aplikace pro prostředí IMS na systému z/OS” na stránce 1138
- “Ukázky vložení na z/OS” na stránce 1139
- “Získat ukázky na webu z/OS” na stránce 1140
- “Ukázka procházení na z/OS” na stránce 1143
- “Ukázka Tisk zprávy na webu z/OS” na stránce 1145
- “Ukázka Atributy fronty na systému z/OS” na stránce 1149
- “Ukázka Správce pošty na webu z/OS” na stránce 1150
- “Ukázka kontroly kreditu na z/OS” na stránce 1156
- “Ukázka obslužné rutiny zpráv v adresáři z/OS” na stránce 1166
- “Ukázka asynchronního vložení na z/OS” na stránce 1170
- “Ukázka Dávková asynchronní spotřeba na systému z/OS” na stránce 1171
- “Ukázka CICS Asynchronní spotřeba a publikování/odběr na z/OS” na stránce 1173
- “Ukázka publikování/odběru na webu z/OS” na stránce 1175
- “Ukázka vlastnosti zprávy Set and Inquire v systému z/OS” na stránce 1177

Související úlohy

“Použití ukázkových programů na platformě Multiplatforms” na stránce 1024
Tyto ukázkové procedurální programy jsou dodávány s produktem. Ukázky jsou napsány v jazycích C a COBOL a ukazují typické použití rozhraní MQI (Message Queue Interface).

z/OS *Funkce demonstrované v ukázkových aplikacích pro z/OS*

Tento oddíl shrnuje funkce rozhraní MQI demonstrované v jednotlivých ukázkových aplikacích, zobrazuje programovací jazyky, v nichž je každá ukázka napsána, a prostředí, v němž je každá ukázka spuštěna.

z/OS *Vložit ukázky na z/OS*

Ukázky Vložit demonstrují, jak vložit zprávy do fronty pomocí volání MQPUT.

Aplikace používá tato volání MQI:

- MQCONN
- MQOPEN
- MQPUT
- MQCLOSE
- MQDISC

Program je dodáván v jazycích COBOL a C a spouští se v dávkovém prostředí a prostředí CICS . Viz Tabulka 172 na stránce 1130 pro dávkovou aplikaci a Tabulka 179 na stránce 1135 pro aplikaci CICS .

z/OS *Získat ukázky na z/OS*

Ukázky Get demonstrují, jak získat zprávy z fronty pomocí volání MQGET.

Aplikace používá tato volání MQI:

- MQCONN
- MQOPEN
- MQGET
- MQCLOSE
- MQDISC

Program je dodáván v jazycích COBOL a C a spouští se v dávkovém prostředí a prostředí CICS . Viz Tabulka 172 na stránce 1130 pro dávkovou aplikaci a Tabulka 179 na stránce 1135 pro aplikaci CICS .

z/OS *Procházet ukázku na z/OS*

Ukázka Procházet demonstruje, jak pomocí volby Procházet vyhledat zprávu, vytisknout ji a poté procházet zprávy ve frontě.

Aplikace používá tato volání MQI:

- MQCONN
- MQOPEN
- MQGET pro procházení zpráv
- MQCLOSE
- MQDISC

Program je dodáván v jazycích COBOL, assembler, PL/I a C. Aplikace se spouští v dávkovém prostředí. Viz Tabulka 173 na stránce 1131 pro dávkovou aplikaci.

z/OS *Vytisknout ukázku zprávy na z/OS*

Ukázka Tisk zprávy demonstruje, jak odebrat zprávu z fronty a vytisknout data ve zprávě spolu se všemi poli jejího deskriptoru zprávy. Volitelně může zobrazit všechny vlastnosti zprávy přidružené ke každé zprávě.

Odebráním znaků komentáře ze dvou řádků ve zdrojovém modulu můžete změnit program tak, aby procházal zprávy ve frontě, a nikoli odebíráním. Tento program lze použít k diagnostice 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
- MQCRTMH
- MQDLTMH
- MQINQMP

Program je dodáván v jazyce C. Aplikace se spouští v dávkovém prostředí. Viz Tabulka 174 na stránce 1131 pro dávkovou aplikaci.

z/OS *Ukázka atributů fronty na z/OS*

Ukázka Atributy fronty demonstruje, jak zjistit 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 spouští v prostředí CICS . Viz Tabulka 180 na stránce 1135 pro aplikaci CICS .

z/OS *Ukázka správce pošty na z/OS*

Aspekty, které je třeba vzít na vědomí při použití ukázky Správce pošty.

Ukázka Správce pošty demonstruje tyto techniky:

- Použití front aliasů
- Použití modelové fronty k vytvoření dočasné dynamické fronty
- Použití front pro odpovědi
- Použití synchronizovaných bodů v prostředí CICS a dávkových prostředích
- Odesílání příkazů do vstupní fronty systémových příkazů
- Testování návratových kódů
- Odesílání zpráv do vzdálených správců 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 IBM MQ for z/OS a obsahují některé panely ISPF .

Viz Tabulka 177 na stránce 1133 pro aplikaci TSO a Tabulka 181 na stránce 1136 pro aplikaci CICS .

z/OS *Ukázka kreditní kontroly na z/OS*

Tyto informace obsahují body, které je třeba zvážit při použití vzorku kontroly kreditu.

Ukázka kontroly kreditu je sada programů, která demonstruje tyto techniky:

- Vývoj 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 a perzistence zpráv
- Spuštění programů pomocí spouštění
- Použití front pro odpovědi
- Použití front aliasů
- Použití fronty nedoručených zpráv
- Použití seznamu názvů
- Testování návratových kódů

Aplikace používá tato volání MQI:

- MQOPEN
- MQPUT
- MQPUT1

- Příkaz MQGET pro procházení a získávání zpráv, použití voleb čekání a signálu a pro získání specifické zprávy.
- MQINQ
- MQSET
- MQCLOSE

Ukázku lze spustit jako samostatnou aplikaci CICS . Chcete-li však předvést, jak navrhnout aplikaci pro řazení zpráv do fronty, která používá prostředky poskytované prostředím CICS i IMS , je jeden modul také dodáván jako program pro dávkové zpracování zpráv IMS .

Programy CICS jsou dodávány v jazycích C a COBOL. Jediný program IMS je dodáván v jazyce C.

Viz Tabulka 182 na stránce 1136 pro aplikaci CICS a Tabulka 184 na stránce 1138 pro aplikaci IMS .

z/OS *Ukázka obslužné rutiny zpráv v adresáři z/OS*

Ukázka obslužné rutiny zpráv vám 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 programovacích jazycích C a COBOL. Aplikace je spuštěna pod TSO. VizTabulka 178 na stránce 1134pro aplikaci TSO.

z/OS *Ukázky ukončení distribuovaných front v systému z/OS*

Tabulka zdrojových programů distribuovaných ukázek ukončení fronty.

Názvy zdrojových programů distribuovaných ukázek uživatelské procedury fronty jsou uvedeny v následující tabulce:

Tabulka 170. Zdroj pro ukázky ukončení distribuovaných front

Název člena	Pro jazyk	Popis	Dodáváno v knihovně
CSQ4BAX0	Sestavovací modul	Zdrojový program	SCSQASMS
CSQ4BCX1	C	Zdrojový program	SCSQC37S
CSQ4BCX2	C	Zdrojový program	SCSQC37S
CSQ4BCX4	C	Zdrojový program	SCSQC37S

Poznámka: Zdrojové programy jsou upravovány pomocí příkazu CSQXSTUB.

z/OS *Ukázky ukončení převodu dat na z/OS*

Kostra je poskytnuta pro uživatelskou proceduru převodu dat a ukázka je dodána s produktem IBM MQ , který ilustruje volání MQXCNVC.

Názvy zdrojových programů ukázek uživatelské procedury pro převod dat jsou uvedeny v následující tabulce:

Tabulka 171. Zdroj pro ukázky uživatelské procedury převodu dat (pouze jazyk assembleru)		
Název člena	**Popis**	**Dodáváno v knihovně**
CSQ4BAX8	Zdrojový program	SCSQASMS
CSQ4BAX9	Zdrojový program	SCSQASMS
CSQ4CAX9	Zdrojový program	SCSQASMS

Poznámka: Zdrojové programy jsou linkovány s CSQASTUB.

Další informace viz “Zápis uživatelských procedur převodu dat” na stránce 950.

z/OS *Ukázky publikování/odběru na webu z/OS*

Ukázkové programy publikování/odběru demonstrují použití funkcí publikování a odběru v produktu IBM MQ.

Existují čtyři ukázkové programy programovacího jazyka C a dva programy programovacího jazyka COBOL, které demonstrují, jak programovat do rozhraní publikování/odběru IBM MQ .

Aplikace používají tato volání MQI:

- MQCONN
- MQOPEN
- MQPUT
- MQSUB
- MQGET
- MQCLOSE
- MQDISC
- MQCRTMH
- MQDLTMH
- MQINQMP

Ukázkové programy Public/Subscribe jsou dodávány v programovacích jazycích C a COBOL. Ukázkové aplikace se spouštějí v dávkovém prostředí. Viz Ukázky publikování/odběru pro dávkové aplikace.

z/OS *Konfigurace správce front pro přijetí připojení klienta v systému z/OS*

Než budete moci spustit ukázkové aplikace, musíte nejprve vytvořit správce front. Poté můžete nakonfigurovat správce front tak, aby bezpečně přijímal příchozí požadavky na připojení od aplikací spuštěných v režimu klienta.

Než začnete

Ujistěte se, že správce front již existuje a byl spuštěn. Zadáním příkazu MQSC určete, zda jsou záznamy ověřování kanálu již povoleny:

```
DISPLAY QMGR CHLAUTH
```

Důležité: Tato úloha očekává, že budou povoleny záznamy ověřování kanálu. Pokud se jedná o správce front používaného jinými uživateli a aplikacemi, změna tohoto nastavení ovlivní všechny ostatní uživatele a aplikace. Pokud váš správce front nepoužívá záznamy ověřování kanálu, lze krok 4 nahradit alternativní metodou ověřování (například uživatelskou procedurou pro zabezpečení zprávy), která nastaví uživatele MCAUSER na *non-privileged-user-id* , kterou získáte v kroku “1” na stránce 1128.

Musíte vědět, který název kanálu vaše aplikace očekává, že jej bude používat, aby aplikace mohla kanál používat. Musíte také vědět, které objekty, například fronty nebo témata, které vaše aplikace očekává, aby je mohla používat.

Informace o této úloze

Tato úloha vytvoří neprivilegované ID uživatele, které má být použito pro klientskou aplikaci, která se připojuje ke správci front. Klientská aplikace má přístup pouze k tomu, aby mohla používat kanál, který potřebuje, a frontu, kterou potřebuje, pomocí tohoto ID uživatele.

Postup

1. Získejte ID uživatele v systému, ve kterém je spuštěn váš správce front.

   Pro tuto úlohu nesmí být toto ID uživatele privilegovaným administrativním uživatelem. Toto ID uživatele je oprávnění, pod kterým bude připojení klienta spuštěno ve správci front.
2. Spusťte program modulu listener.
   a) Ujistěte se, že je inicializátor kanálu spuštěn. Pokud ne, spusťte jej zadáním příkazu **START CHINIT** .
   b) Spusťte program listener zadáním následujícího příkazu:

   ```
   START LISTENER TRPTYPE(TCP) PORT(nnnn)
   ```

   kde *nnnn* je zvolené číslo portu.
3. Pokud vaše aplikace používá systém SYSTEM.DEF.SVRCONN pak 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. Zadáním příkazu MQSC vytvořte pravidlo ověřování kanálu, které umožní kanálu používat pouze adresu IP klientského systému:

   ```
   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. Pokud je ukázková klientská aplikace spuštěna na stejném počítači jako správce front, použijte adresu IP '127.0.0.1', pokud se vaše aplikace bude připojovat pomocí 'localhost'. Pokud se bude připojovat několik různých klientských počítačů, můžete místo jediné adresy IP použít vzor nebo rozsah. Podrobnosti viz Generické adresy IP .
   non-privileged-user-id je ID uživatele, které jste získali v kroku “1” na stránce 1128
5. Pokud vaše aplikace používá systém 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 vaší fronty.
6. Udělte přístup pro připojení a dotaz na správce front:
   a) Ujistěte se, že je inicializátor kanálu spuštěn. Pokud ne, spusťte inicializátor kanálu zadáním příkazu START CHINIT.
   b) Spusťte modul listener TCP, například zadejte následující příkaz:

   ```
   START LISTENER TRPTYPE(TCP) PORT(nnnn)
   ```

   kde *nnnn* je zvolené číslo portu.

7. Pokud je vaše aplikace dvoubodovou aplikací, tj. používá fronty, udělte přístup, abyste umožnili dotazování a vkládání a získávání zpráv pomocí vaší fronty pomocí ID uživatele, které má být použito, zadáním 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 správce front.
   název-fronty je název vaší fronty.
   non-privileged-user-id je ID uživatele, které jste získali v kroku “1” na stránce 1128
8. Pokud je vaše aplikace aplikací typu publikování/odběr, tj. používá témata, udělte přístup, abyste povolili publikování a přihlášení k odběru vašeho tématu pomocí ID uživatele, které se má použít, zadáním následujících příkazů 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 správce front.
   non-privileged-user-id je ID uživatele, které jste získali v kroku “1” na stránce 1128
   To umožní *non-privileged-user-id* přístup k libovolnému tématu ve stromu témat, případně můžete definovat objekt tématu pomocí **DEFINE TOPIC** a udělit přístup pouze k části stromu témat, na kterou odkazuje daný objekt tématu. Další informace naleznete v tématu Řízení přístupu uživatelů k tématům.

Jak pokračovat dále

Aplikace klienta se nyní může připojit ke správci front a vkládat nebo získávat zprávy pomocí fronty.

Související pojmy

z/OS Oprávnění pro práci s objekty IBM MQ na systému z/OS

Související odkazy

NASTAVIT CHLAUTH
Definovat kanál
DEFINICE QLOCAL
SET AUTHREC

z/OS *Příprava a spuštění ukázkových aplikací pro dávkové prostředí v systému z/OS*

Chcete-li připravit ukázkovou aplikaci, která se spustí v dávkovém prostředí, proveďte stejné kroky, které byste provedli při sestavování libovolné dávkové aplikace IBM MQ for z/OS .

Tyto kroky jsou uvedeny v části “Sestavení dávkových aplikací z/OS” na stránce 990.

Případně, pokud dodáme spustitelnou formu ukázky, můžete ji spustit z zaváděcí knihovny thlqual.SCSQLOAD .

Poznámka: Jazyková verze assembleru ukázky Procházet používá řídicí bloky dat (DCB), takže ji musíte propojit-upravit pomocí RMODE(24).

Členové knihovny, kteří mají být použiti, jsou uvedeni v adresáři Tabulka 172 na stránce 1130, Tabulka 173 na stránce 1131, Tabulka 174 na stránce 1131a Tabulka 175 na stránce 1131.

Musíte upravit skript JCL dodaný pro ukázky, které chcete použít (viz Tabulka 172 na stránce 1130, Tabulka 173 na stránce 1131, Tabulka 174 na stránce 1131a Tabulka 175 na stránce 1131).

Příkaz PARM v dodaném JCL obsahuje řadu parametrů, které je třeba upravit. Chcete-li spustit ukázkové programy v jazyce C, oddělte parametry mezerami; chcete-li spustit ukázkový program assembleru, COBOL a PL/I, oddělte je čárkami. Pokud je například název vašeho správce front CSQ1 a chcete spustit aplikaci s frontou s názvem LOCALQ1v jazyce JCL v jazycích COBOL, PL/I a assembler, měl by váš příkaz PARM vypadat takto:

```
PARM=(CSQ1,LOCALQ1)
```

V JCL v jazyce C by měl váš příkaz PARM vypadat takto:

```
PARM=('CSQ1 LOCALQ1')
```

Nyní jste připraveni odeslat úlohy.

z/OS *Názvy ukázkových dávkových aplikací na systému z/OS*
Souhrn programů, které jsou dodávány pro ukázkové dávkové aplikace.

Dávkové aplikační programy jsou shrnuty v následujících tabulkách:

- Tabulka 172 na stránce 1130 Vložit a získat ukázky
- Tabulka 173 na stránce 1131 Procházet ukázku
- Tabulka 174 na stránce 1131 Vytisknout ukázku zprávy
- Tabulka 175 na stránce 1131 Ukázky publikování/odběru
- Tabulka 176 na stránce 1132 Další ukázky

Tabulka 172. Dávkové vkládání a získávání vzorků

Název člena	**Pro jazyk**	**Popis**	**Zdrojový soubor dodaný v knihovně**	**Spustitelný soubor dodaný v knihovně**
CSQ4BCJ1	C	Získat zdrojový program	SCSQC37S	SCSQLOAD
CSQ4BCK1	C	Vložit zdrojový program	SCSQC37S	SCSQLOAD
CSQ4BCJR	C	Ukázka spuštění JCL pro CSQ4BCJ1 a CSQBCK1	SCSQPROC	Není
CSQ4BVJ1	COBOL	Získat zdrojový program	SCSQCOBS	SCSQLOAD
CSQ4BVK1	COBOL	Vložit zdrojový program	SCSQCOBS	SCSQLOAD
CSQ4BVJR	COBOL	Ukázka spuštění JCL pro CSQBVJ1 a CSQBVK1	SCSQPROC	Není

Tabulka 173. Ukázka dávkového procházení

Název člena	Pro jazyk	Popis	Zdrojový soubor dodaný v knihovně	Spustitelný soubor dodaný v knihovně
CSQ4BVA1	COBOL	Zdrojový program	SCSQCOBS	SCSQLOAD
CSQ4BVAR	COBOL	Ukázka spuštění JCL pro CSQ4BVA1	SCSQPROC	Není
CSQ4BAA1	Sestavovací modul	Zdrojový program	SCSQASMS	SCSQLOAD
CSQ4BAAR	Sestavovací modul	Ukázka spuštění JCL pro CSQ4BAA1	SCSQPROC	Není
CSQ4BCA1	C	Zdrojový program	SCSQC37S	SCSQLOAD
CSQ4BCAR	C	Ukázka spuštění JCL pro CSQ4BCA1	SCSQPROC	Není
CSQ4BPA1	PL/I	Zdrojový program	SCSQPLIS	SCSQLOAD
CSQ4BPAR	PL/I	Ukázka spuštění JCL pro CSQ4BPA1	SCSQPROC	Není

Tabulka 174. Ukázka dávkového tisku zprávy (pouze jazyk C)

Název člena	Popis	Zdrojový soubor dodaný v knihovně	Spustitelný soubor dodaný v knihovně
CSQ4BCG1	Zdrojový program	SCSQC37S	SCSQLOAD
CSQ4BCGR	Ukázka spuštění JCL pro CSQ4BCG1	SCSQPROC	Není
CSQ4BCL1	Procházet zdrojový program	SCSQC37S	SCSQLOAD
CSQ4BCLR	Ukázka spuštění JCL pro CSQ4BCL1	SCSQPROC	Není

Tabulka 175. 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	SCSQLOAD
CSQ4BCP2	C	Přihlásit se k odběru tématu a získat zdrojový program zpráv	SCSQC37S	CSQ4BCPS	SCSQLOAD
CSQ4BCP3	C	Přihlásit se k odběru tématu pomocí uživatelem poskytnutého místa určení a získat zdrojový program zpráv	SCSQC37S	CSQ4BCPD	SCSQLOAD

Tabulka 175. Ukázky publikování/odběru (pokračování)

Název člena	Pro jazyk	Popis	Zdrojový soubor dodaný v knihovně	JCL v SCSQPROC	Spustitelný soubor dodaný v knihovně
CSQ4BCP4	C	Přihlásit se k odběru tématu pomocí rozšířených voleb a získat zdrojový program zpráv	SCSQC37S	CSQ4BCPE	SCSQLOAD
CSQ4BVP1	COBOL	Publikovat do zdrojového programu tématu	SCSQCOBS	CSQ4BVPP	SCSQLOAD
CSQ4BVP2	COBOL	Přihlásit se k odběru tématu a získat zdrojový program zpráv	SCSQCOBS	CSQ4BVPS	SCSQLOAD

Tabulka 176. Další ukázky

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	SCSQLOAD
CSQ4BCS2	C	Zdrojový program asynchronního vkládání a kontroly stavu	SCSQC37S	CSQ4BCSP	SCSQLOAD
CSQ4BCM1	C	Zdrojový program pro dotazování vlastností zprávy	SCSQC37S	CSQ4BCMP	SCSQLOAD
CSQ4BCM2	C	Nastavit zdrojový program vlastností zprávy	SCSQC37S	CSQ4BCMP	SCSQLOAD

z/OS *Příprava ukázkových aplikací pro prostředí TSO na systému z/OS*

Chcete-li připravit ukázkovou aplikaci, která se spustí v prostředí TSO, proveďte stejné kroky, které byste provedli při sestavování libovolné dávkové aplikace IBM MQ for z/OS .

Tyto kroky jsou uvedeny v části “Sestavení dávkových aplikací z/OS” na stránce 990. Členy knihovny, které se mají použít, jsou uvedeny v seznamu Tabulka 177 na stránce 1133.

Případně, pokud dodáme spustitelnou formu ukázky, můžete ji spustit z zaváděcí knihovny thlqual.SCSQLOAD .

Pro ukázkovou aplikaci Mail Manager se ujistěte, že fronty, které používá, jsou v systému k dispozici. Jsou definovány ve členu **thlqual**.SCSQPROC (CSQ4CVD). Chcete-li zajistit, aby byly tyto fronty vždy k dispozici, můžete tyto členy přidat do datové sady vstupu inicializace CSQINP2 nebo tyto definice front načíst pomocí programu CSQUTIL.

z/OS *Názvy ukázkových aplikací TSO na systému z/OS*

Informace o názvech programů, které jsou dodávány pro každou ukázkovou aplikaci TSO, a o knihovnách, kde se nachází zdroj, JCL a pro ukázku obslužné rutiny zpráv pouze spustitelné soubory.

Aplikační programy TSO jsou shrnuty v následujících tabulkách:

- Tabulka 177 na stránce 1133 Ukázka správce pošty
- Tabulka 178 na stránce 1134 Ukázková obslužná rutina zpráv

Tyto ukázky používají panely ISPF . Proto musíte při vytváření odkazů na programy zahrnout stub ISPF ISPLINK.

Tabulka 177. Ukázka Správce pošty TSO

Název člena	**Pro jazyk**	**Popis**	**Zdrojový soubor dodaný v knihovně**
CSQ4CVD	Nezávislý	Definice objektů IBM MQ for z/OS	SCSQPROC
CSQ40	Nezávislý	Zprávy ISPF	SCSQMSGE
CSQ4RVD1	COBOL	CLIST pro zahájení CSQ4TVD1	SCSQCLST
CSQ4TVD1	COBOL	Zdrojový program pro program Menu	SCSQCOBS
CSQ4TVD2	COBOL	Zdrojový program pro program Get Mail	SCSQCOBS
CSQ4TVD4	COBOL	Zdrojový program pro program Odeslat poštu	SCSQCOBS
CSQ4TVD5	COBOL	Zdrojový program pro program Nickname	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 inicializaci CSQ4TCD1	SCSQCLST
CSQ4TCD1	C	Zdrojový program pro program Menu	SCSQC37S
CSQ4TCD2	C	Zdrojový program pro program Get Mail	SCSQC37S
CSQ4TCD4	C	Zdrojový program pro program Odeslat poštu	SCSQC37S
CSQ4TCD5	C	Zdrojový program pro program Nickname	SCSQC37S
CSQ4CDP1-6	C	Definice panelů	SCSQPNLA
CSQ4TC0	C	Zahrnout soubor	SCSQC370

Tabulka 178. Ukázka obslužné rutiny 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	SCSQLOAD
CSQ4TCH2	C	Zdrojový program	SCSQC37S	SCSQLOAD
CSQ4TCH3	C	Zdrojový program	SCSQC37S	SCSQLOAD
CSQ4RCH1	Jazyk C a COBOL	CLIST pro inicializaci CSQ4TCH1 nebo CSQ4TVH1	SCSQCLST	Není
CSQ4CHP1	Jazyk C a COBOL	Definice panelu	SCSQPNLA	Není
CSQ4CHP2	Jazyk C a COBOL	Definice panelu	SCSQPNLA	Není
CSQ4CHP3	Jazyk C a COBOL	Definice panelu	SCSQPNLA	Není
CSQ4CHP9	Jazyk C a COBOL	Definice panelu	SCSQPNLA	Není
CSQ4TVH0	COBOL	definice dat	SCSQCOBC	Není
CSQ4TVH1	COBOL	Zdrojový program	SCSQCOBS	SCSQLOAD
CSQ4TVH2	COBOL	Zdrojový program	SCSQCOBS	SCSQLOAD
CSQ4TVH3	COBOL	Zdrojový program	SCSQCOBS	SCSQLOAD

z/OS *Příprava ukázkových aplikací pro prostředí CICS na systému z/OS*

Před spuštěním ukázkových programů CICS se přihlaste do systému CICS pomocí parametru LOGMODE 32702. Důvodem je, že ukázkové programy byly napsány tak, aby používaly obrazovku režimu 3270 2.

Chcete-li připravit ukázkovou aplikaci, která se spustí v prostředí CICS , postupujte takto:

1. Vytvořte symbolickou mapu popisu a fyzickou mapu obrazovky pro ukázku sestavením zdroje definice obrazovky BMS (dodávaného v knihovně **thlqual**.SCSQMAPS, kde **thlqual** je kvalifikátor vyšší úrovně používaný vaší instalací). Když pojmenujete mapy, použijte název zdroje definice obrazovky BMS (není k dispozici pro ukázkové programy Put a Get), ale vynechte poslední znak tohoto názvu.
2. Proveďte stejné kroky, které byste provedli při sestavování libovolné aplikace CICS IBM MQ for z/OS . Tyto kroky jsou uvedeny v části “Sestavení aplikací CICS v produktu z/OS” na stránce 993. Členové knihovny, kteří mají být použiti, jsou uvedeni v adresáři Tabulka 179 na stránce 1135, Tabulka 180 na stránce 1135, Tabulka 181 na stránce 1136a Tabulka 182 na stránce 1136.

   Případně, pokud dodáme spustitelnou formu ukázky, můžete ji spustit z zaváděcí knihovny thlqual.SCSQCICS .
3. Identifikujte sadu map, programy a transakce na CICS aktualizací datové sady CSD (CICS system definition). Požadované definice jsou ve členu **thlqual**.SCSQPROC (CSQ4S100). Pokyny, jak to provést, naleznete v části *Adaptér CICS-IBM MQ* v dokumentaci k produktu CICS Transaction Server for z/OS 4.1 na adrese: CICS Transaction Server for z/OS 4.1, The CICS-IBM MQ adapter.

   Poznámka: Pro ukázkovou aplikaci Credit Check se v této fázi zobrazí chybová zpráva, pokud jste dosud nevytvořili datovou sadu VSAM, kterou ukázka používá.
4. V případě ukázkových aplikací Credit Check a Mail Manager se ujistěte, že fronty, které používají, jsou v systému k dispozici. Pro ukázku kontroly kreditu jsou definovány ve členu **thlqual**.SCSQPROC (CSQ4CVB) pro COBOL a **thlqual**.SCSQPROC (CSQ4CCB) pro C. Pro ukázku správce pošty jsou definovány ve členu **thlqual**.SCSQPROC (CSQ4CVD). Chcete-li zajistit, aby byly tyto fronty vždy

k dispozici, můžete tyto členy přidat do datové sady vstupu inicializace CSQINP2 nebo tyto definice front načíst pomocí programu CSQUTIL.

Pro ukázkovou aplikaci Atributy fronty můžete použít jednu nebo více front, které jsou dodávány pro ostatní ukázkové aplikace. Případně můžete použít vlastní fronty. Avšak ve formě, která je dodána, tato ukázka pracuje pouze s frontami, které mají znaky CSQ4SAMP v prvních osmi bajtech svého názvu.

z/OS *Názvy ukázkových aplikací CICS na systému z/OS*

Toto téma obsahuje souhrn programů dodávaných pro ukázkové aplikace CICS .

Aplikační programy CICS jsou shrnuty v následujících tabulkách:

- Tabulka 179 na stránce 1135 Vložit a získat ukázky
- Ukázka atributů fronty Tabulka 180 na stránce 1135
- Tabulka 181 na stránce 1136 Mail Manager, ukázka (pouze COBOL)
- Tabulka 182 na stránce 1136 Ukázka kontroly kreditu
- Tabulka 183 na stránce 1137 Ukázky asynchronní spotřeby a publikování/odběru

Tabulka 179. CICS Vložit 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 180. Ukázka atributů fronty CICS

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	SCSQMAPS-mapy	SCSQCICS (s názvem CSQ4ACM)
CSQ4CAC1	Sestavovací modul	Zdrojový program	SCSQASMS	SCSQCICS
CSQ4AMSG	Sestavovací modul	Definice zprávy	SCSQMACS	Není
CSQ4ACMS	Sestavovací modul	Definice obrazovky BMS	SCSQMAPS-mapy	SCSQCICS (s názvem CSQ4ACM)
CSQ4CCC1	C	Zdrojový program	SCSQC37S	SCSQCICS
CSQ4CMSG	C	Definice zprávy	SCSQC370	Není

Tabulka 180. Ukázka atributů fronty CICS (pokračování)

Název člena	Pro jazyk	Popis	Zdrojový soubor dodaný v knihovně	Spustitelný soubor dodaný v knihovně
CSQ4CCMS	C	Definice obrazovky BMS	SCSQMAPS-mapy	SCSQCICS (s názvem CSQ4ACM)
CSQ4S100	Nezávislý	Datová sada definice systému CICS	SCSQPROC	Není

Tabulka 181. CICS Mail Manager, ukázka (pouze COBOL)

Název člena	Popis	Zdrojový soubor dodaný v knihovně
CSQ4CVD	Definice objektů IBM MQ for z/OS	SCSQPROC
CSQ4CVD1	Zdroj pro program Menu	SCSQCOBS
CSQ4CVD2	Zdroj pro program Get Mail	SCSQCOBS
CSQ4CVD3	Zdroj pro program Zobrazení zprávy	SCSQCOBS
CSQ4CVD4	Zdroj pro program Odeslat poštu	SCSQCOBS
CSQ4CVD5	Zdroj pro program Přezdívka	SCSQCOBS
CSQ4VDMS	Zdroj definice obrazovky BMS	SCSQMAPS-mapy
CSQ4S100	Datová sada definice systému CICS	SCSQPROC
CSQ4VD0	definice dat	SCSQCOBC
CSQ4VD3	definice dat	SCSQCOBC
CSQ4VD4	definice dat	SCSQCOBC

Tabulka 182. CICS Ukázka kontroly kreditu

Název člena	Pro jazyk	Popis	Zdrojový soubor dodaný v knihovně
CSQ4CVB	Nezávislý	Definice objektů IBM MQ	SCSQPROC
CSQ4CCB	Nezávislý	Definice objektů IBM MQ	SCSQPROC
CSQ4CVB1	COBOL	Zdroj pro program uživatelského rozhraní	SCSQCOBS
CSQ4CVB2	COBOL	Zdroj pro správce žádostí o úvěr	SCSQCOBS
CSQ4CVB3	COBOL	Zdroj pro program pro vrácení účtu	SCSQCOBS
CSQ4CVB4	COBOL	Zdroj pro distribuční program	SCSQCOBS
CSQ4CVB5	COBOL	Zdroj pro program agenturního dotazu	SCSQCOBS
CSQ4CCB1	C	Zdroj pro program uživatelského rozhraní	SCSQC37S
CSQ4CCB2	C	Zdroj pro správce žádostí o úvěr	SCSQC37S

Tabulka 182. CICS Ukázka kontroly kreditu (pokračování)

Název člena	Pro jazyk	Popis	Zdrojový soubor dodaný v knihovně
CSQ4CCB3	C	Zdroj pro program pro vrácení účtu	SCSQC37S
CSQ4CCB4	C	Zdroj pro distribuční program	SCSQC37S
CSQ4CCB5	C	Zdroj pro program agenturního dotazu	SCSQC37S
CSQ4CB0	C	Zahrnout soubor	SCSQC370
CSQ4CBMS	C	Zdroj definice obrazovky BMS	SCSQMAPS-mapy
CSQ4VBMS	COBOL	Zdroj definice obrazovky BMS	SCSQMAPS-mapy
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 používané CSQ4CVB3	SCSQPROC
CSQ4S100	Nezávislý	Datová sada definice systému CICS	SCSQPROC

Tabulka 183. CICS Ukázky asynchronní spotřeby a publikování/odběru

Název člena	Popis	Zdrojový soubor dodaný v knihovně
CSQ4CVCN	Zdroj pro program spotřeby jednoduchých zpráv	SCSQCOBS
CSQ4CVCT	Zdroj pro program Spotřeba řídicích zpráv	SCSQCOBS
CSQ4CVEV	Zdroj pro program obslužné rutiny událostí	SCSQCOBS
CSQ4CVPT	Zdroj pro program klienta vložení zprávy	SCSQCOBS
CSQ4CVRG	Zdroj pro program klienta registrace	SCSQCOBS
CSQ4S100	CICS Datová sada definice systému	SCSQPROC

z/OS *Příprava ukázkové aplikace pro prostředí IMS na systému z/OS*

Část ukázkové aplikace Credit Check lze spustit v prostředí IMS .

Chcete-li připravit tuto část aplikace ke spuštění s ukázkou CICS , proveďte nejprve kroky popsané v části “Příprava ukázkových aplikací pro prostředí CICS na systému z/OS” na stránce 1134.

Poté proveďte následující kroky:

1. Proveďte stejné kroky, které byste provedli při sestavování libovolné aplikace IMS IBM MQ for z/OS . Tyto kroky jsou uvedeny v části “Sestavení aplikací IMS (BMP nebo MPP)” na stránce 994. Členy knihovny, které se mají použít, jsou uvedeny v seznamu Tabulka 184 na stránce 1138.
2. Identifikujte aplikační program a databázi pro IMS. Vzorky jsou poskytovány s příkazy PSBGEN, DBDGEN, ACB, IMSGEN a IMSDALOC, aby to bylo možné.
3. Načtěte databázi CSQ4CA přizpůsobením a spuštěním ukázkového JCL poskytnutého pro tento účel (CSQ4ILDB). Tento JCL načte databázi s daty ze souboru CSQ4BAQ. Aktualizujte řídicí oblast IMS pomocí příkazu DD pro databázi CSQ4CA.
4. Spusťte program checking-account jako program pro dávkové zpracování zpráv (BMP) přizpůsobením a spuštěním ukázkového JCL poskytnutého pro tento účel. Tento skript JCL spouští dávkově orientovaný program BMP. Chcete-li spustit program jako program BMP orientovaný na zprávy, odeberte znaky komentáře z řádku v JCL, který obsahuje příkaz IN=.

z/OS *Názvy ukázkové aplikace IMS v systému z/OS*

Tyto informace poskytují tabulku se seznamem zdrojů a JCL, které jsou dodávány pro ukázkovou aplikaci Credit Check IMS .

Tabulka 184. Zdroj a JCL pro ukázku Credit Check IMS (pouze C)

Název člena	**Popis**	**Dodáváno v knihovně**
CSQ4CVB	Definice objektů IBM MQ	SCSQPROC
CSQ4ICB3	Zdroj pro program pro vrácení účtu	SCSQC37S
CSQ4ICBL	Zdroj pro načtení databáze kontrolních účtů	SCSQC37S
CSQ4CBI	definice dat	SCSQC370
CSQ4PSBL	PSBGEN JCL pro program pro načítání databáze	SCSQPROC
CSQ4PSB3	PSBGEN JCL pro program kontrolních účtů	SCSQPROC
CSQ4DBDS	DBDGEN JCL pro databázi CSQ4CA	SCSQPROC
CSQ4GIMS	IMSDefinice maker GEN pro CSQ4IVB3 a CSQ4CA	SCSQPROC
CSQ4ACBG	Definice řídicího bloku aplikace (ACB) pro CSQ4IVB3	SCSQPROC
CSQ4BAQ	Zdroj pro databázi	SCSQPROC
CSQ4ILDB	Ukázka spuštění JCL pro úlohu načtení databáze	SCSQPROC
CSQ4ICBR	Ukázka spuštění JCL pro program kontrolních účtů	SCSQPROC

Tabulka 184. Zdroj a JCL pro ukázku Credit Check IMS (pouze C) (pokračování)		
Název člena	**Popis**	**Dodáváno v knihovně**
CSQ4DYNA	IMSDefinice maker DALOC pro databázi	SCSQPROC

z/OS *Ukázky vložení na z/OS*

Ukázkové programy Put vložily zprávy do fronty pomocí volání MQPUT.

Zdrojové programy jsou dodávány v jazycích C a COBOL v dávkových prostředích a prostředích CICS (viz Tabulka 172 na stránce 1130 a Tabulka 179 na stránce 1135).

Návrh vzorku pro vložení

Tok logikou programu je:

1. Připojte se ke správci front pomocí volání MQCONN. Pokud se toto volání nezdaří, vytiskněte kódy dokončení a příčiny a ukončete zpracování.

   Poznámka: Pokud spouštíte ukázku v prostředí CICS , nemusíte vydávat volání MQCONN; pokud tak učiníte, vrátí hodnotu DEF_HCONN. Pro následující volání MQI můžete použít manipulátor připojení MQHC_DEF_HCONN.
2. Otevřete frontu pomocí volání MQOPEN s volbou MQOO_OUTPUT. Na vstupu tohoto volání program použije manipulátor připojení, který je vrácen v kroku “1” na stránce 1141. Pro strukturu deskriptoru objektu (MQOD) používá výchozí hodnoty pro všechna pole kromě pole názvu fronty, které je programu předáno jako parametr. Pokud volání MQOPEN selže, vytiskněte kódy dokončení a příčiny a zastavte zpracování.
3. Vytvořte smyčku v rámci programu vydávajícího volání MQPUT, dokud nebude do fronty vložen požadovaný počet zpráv. Pokud volání MQPUT selže, smyčka je předčasně opuštěna, nejsou zkoušena žádná další volání MQPUT a jsou vráceny kódy příčiny a dokončení.
4. Zavřete frontu pomocí volání MQCLOSE s popisovačem objektu vráceným v kroku “2” na stránce 1141. Pokud toto volání selže, vytiskněte kódy dokončení a příčiny.
5. Odpojte se od správce front pomocí volání MQDISC s manipulátoru připojení vráceným v kroku “1” na stránce 1141. Pokud toto volání selže, vytiskněte kódy dokončení a příčiny.

   Poznámka: Pokud spouštíte ukázku v prostředí CICS , nemusíte volat MQDISC.

z/OS *Ukázky Vložit pro dávkové prostředí na systému z/OS*

Toto téma použijte při zvažování vložení ukázek pro dávkové prostředí.

Chcete-li spustit ukázky, upravte a spusťte ukázkový soubor JCL, jak je popsáno v tématu “Příprava a spuštění ukázkových aplikací pro dávkové prostředí v systému z/OS” na stránce 1129.

Programy používají následující parametry v EXEC PARM, oddělené mezerami v C a čárkami v COBOLu:

1. Název správce front (4 znaky)
2. Název cílové fronty (48 znaků)
3. Počet zpráv (až 4 číslice)
4. Výplňový znak, který se má zapsat do zprávy (1 znak)
5. Počet znaků, které se mají zapsat do zprávy (až 4 číslice)
6. Perzistence zprávy (1 znak: P pro trvalou nebo N pro dočasnou)

Zadáte-li některý z těchto parametrů nesprávně, zobrazí se příslušné chybové zprávy.

Všechny zprávy z ukázek se zapisují do datové sady SYSPRINT.

Poznámky k použití

- Aby byly ukázky jednoduché, existují některé drobné funkční rozdíly mezi jazykovými verzemi. Tyto rozdíly jsou však minimalizovány, pokud použijete rozvržení parametrů uvedených v ukázkovém kódu JCL, CSQ4BCJRa CSQ4BVJR. Žádný z rozdílů se netýká rozhraní MQI.
- CSQ4BCK1 umožňuje zadat více než čtyři číslice pro počet odeslaných zpráv a délku zpráv.
- Pro dvě číselná pole zadejte libovolnou číslici v rozsahu 1 až 9999. Hodnota, kterou zadáte, by měla být kladné číslo. Chcete-li například vložit jedinou zprávu, můžete jako hodnotu zadat 1, 01, 001 nebo 0001. Zadáte-li nečíselné nebo záporné hodnoty, může dojít k chybě. Zadáte-li například -1, program v jazyce COBOL odešle jednobajtovou zprávu, ale program v jazyce C obdrží chybu.
- Pro oba programy, CSQ4BCK1 a CSQ4BVK1, musíte zadat P do parametru perzistence, + + PER + +, pokud chcete, aby byla zpráva trvalá. Pokud tak neučiníte, bude zpráva dočasná.

z/OS *Ukázky Vložit pro prostředí CICS na systému z/OS*

Toto téma použijte při zvažování ukázek Vložit pro prostředí CICS .

Transakce provádějí následující parametry oddělené čárkami:

1. Počet zpráv (až 4 číslice)
2. Výplňový znak, který se má zapsat do zprávy (1 znak)
3. Počet znaků, které se mají zapsat do zprávy (až 4 číslice)
4. Perzistence zprávy (1 znak: P pro trvalou nebo N pro dočasnou)
5. Název cílové fronty (48 znaků)

Zadáte-li některý z těchto parametrů nesprávně, obdržíte příslušné chybové zprávy.

V případě ukázky COBOL vyvolejte ukázku Put v prostředí CICS zadáním následujícího příkazu:

```
MVPT,9999,*,9999,P,QUEUE.NAME
```

Pro ukázku C vyvolejte ukázku Put v prostředí CICS zadáním následujícího příkazu:

```
MCPT,9999,*,9999,P,QUEUE.NAME
```

Na obrazovce se zobrazí všechny zprávy z ukázek.

Poznámky k použití

- Aby byly ukázky jednoduché, existují některé drobné funkční rozdíly mezi jazykovými verzemi. Žádný z rozdílů se netýká rozhraní MQI.
- Zadáte-li název fronty delší než 48 znaků, bude jeho délka oříznuta na maximální délku 48 znaků, ale nebude vrácena žádná chybová zpráva.
- Před zadáním transakce stiskněte klávesu CLEAR.
- Pro dvě číselná pole zadejte libovolné číslo v rozsahu 1 až 9999. Hodnota, kterou zadáte, by měla být kladné číslo. Chcete-li například vložit jedinou 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 se ukončí s chybou malloc ().
- Pro oba programy, CSQ4CCK1 a CSQ4CVK1, zadejte P do parametru perzistence, chcete-li, aby byla zpráva trvalá. V případě dočasných zpráv zadejte do parametru perzistence hodnotu N. Pokud zadáte jinou hodnotu, obdržíte chybovou zprávu.
- Zprávy jsou vloženy do synchronizačního bodu, protože výchozí hodnoty jsou použity pro všechny parametry kromě těch, které byly nastaveny během vyvolání programu.

z/OS ***Získat ukázky na webu z/OS***

Ukázkové programy Get získají zprávy z fronty pomocí volání MQGET.

Zdrojové programy jsou dodávány v jazycích C a COBOL v dávkových prostředích a prostředích CICS (viz Tabulka 172 na stránce 1130 a Tabulka 179 na stránce 1135).

z/OS *Návrh vzorku Get na z/OS*

Seznamte se s návrhem ukázky Get a s poznámkami k použití, které je třeba zvážit.

Tok logikou programu je:

1. Připojte se ke správci front pomocí volání MQCONN. Pokud se toto volání nezdaří, vytiskněte kódy dokončení a příčiny a ukončete zpracování.

   Poznámka: Pokud spouštíte ukázku v prostředí CICS , nemusíte vydávat volání MQCONN; pokud tak učiníte, vrátí hodnotu DEF_HCONN. Pro následující volání MQI můžete použít manipulátor připojení MQHC_DEF_HCONN.
2. Otevřete frontu pomocí volání MQOPEN s volbami MQOO_INPUT_SHARED a MQOO_BROWSE. Na vstupu tohoto volání program použije manipulátor připojení, který je vrácen v kroku "1" na stránce 1141. Pro strukturu deskriptoru objektu (MQOD) používá výchozí hodnoty pro všechna pole kromě pole názvu fronty, které je programu předáno jako parametr. Pokud volání MQOPEN selže, vytiskněte kódy dokončení a příčiny a zastavte zpracování.
3. Vytvořte smyčku v rámci programu vydávajícího volání MQGET, dokud nebude z fronty načten požadovaný počet zpráv. Dojde-li k selhání volání MQGET, smyčka bude předčasně opuštěna, nebude proveden žádný další pokus o volání MQGET a budou vráceny kódy dokončení a příčiny. Ve volání MQGET jsou určeny následující volby:

   - MQGMO_NO_WAIT
   - MQGMO_ACCEPT_TRUNCATED_MESSAGE
   - MQGMO_SYNCPOINT nebo MQGMO_NO_SYNCPOINT
   - MQGMO_BROWSE_FIRST a MQGMO_BROWSE_NEXT

   Popis těchto voleb viz MQGET. Pro každou zprávu se vytiskne číslo zprávy následované délkou zprávy a daty zprávy.
4. Zavřete frontu pomocí volání MQCLOSE s popisovačem objektu vráceným v kroku "2" na stránce 1141. Pokud toto volání selže, vytiskněte kódy dokončení a příčiny.
5. Odpojte se od správce front pomocí volání MQDISC s manipulátoru připojení vráceným v kroku "1" na stránce 1141. Pokud toto volání selže, vytiskněte kódy dokončení a příčiny.

   Poznámka: Pokud spouštíte ukázku v prostředí CICS , nemusíte volat MQDISC.

Poznámky k použití

- Aby byly ukázky jednoduché, existují některé drobné funkční rozdíly mezi jazykovými verzemi. Tyto rozdíly jsou však minimalizovány, pokud použijete rozvržení parametrů uvedených v ukázkovém kódu JCL, CSQ4BCJRa CSQ4BVJR. Žádný z rozdílů se netýká rozhraní MQI.
- CSQ4BCJ1 umožňuje zadat více než čtyři číslice pro počet načtených zpráv.
- Zprávy delší než 64 kB jsou oříznuty.
- CSQ4BCJ1 může správně zobrazit pouze znakové zprávy, protože se zobrazuje pouze do zobrazení prvního znaku NULL (\0).
- Do pole číselný počet zpráv zadejte libovolnou číslici v rozsahu 1 až 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 -1, program v jazyce COBOL načte jednu zprávu, ale program v jazyce C žádné zprávy nenačte.
- 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 synchronizačního bodu, + + SYNC + +, pro zprávy, které mají být načteny v synchronizačním bodu.

z/OS *Ukázky Get pro dávkové prostředí na systému z/OS*

Chcete-li spustit ukázky, upravte a spusťte ukázkový soubor JCL, jak je popsáno v tématu “Příprava a spuštění ukázkových aplikací pro dávkové prostředí v systému z/OS” na stránce 1129.

Programy používají následující parametry v EXEC PARM, oddělené mezerami v C a čárkami v COBOLu:

1. Název správce front (4 znaky)
2. Název cílové fronty (48 znaků)
3. Počet zpráv, které se mají získat (až 4 číslice)
4. Volba procházení/získání zprávy (1 znak: B pro procházení nebo D pro destruktivní 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 Get pro prostředí CICS na systému z/OS*

Speciální aspekty pro získání ukázek pro prostředí CICS .

Transakce provádějí v modulu EXEC PARM následující parametry oddělené čárkami:

1. Počet zpráv, které se mají získat (až čtyři číslice)
2. Volba procházení/získání zprávy (jeden znak: B pro procházení nebo D pro destruktivní 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.

V případě ukázky COBOL vyvolejte ukázku Get v prostředí CICS zadáním následujícího příkazu:

```
MVGT,9999,B,S,QUEUE.NAME
```

V případě ukázky C vyvolejte ukázku Get v prostředí CICS zadáním následujícího příkazu:

```
MCGT,9999,B,S,QUEUE.NAME
```

Když jsou zprávy načteny z fronty, jsou vloženy do fronty dočasného úložiště CICS se stejným názvem jako transakce CICS (například MCGT pro ukázku C).

Zde je příklad výstupu ukázek Get:

```
*************************** TOP OF QUEUE ************************
000000000 : 000000010: **********
```

```
000000001 : 000000010 :**********
************************** BOTTOM OF QUEUE **********************
```

Poznámky k použití

- Aby byly ukázky jednoduché, existují některé drobné funkční rozdíly mezi jazykovými verzemi. Žádný z rozdílů se netýká rozhraní MQI.
- Zadáte-li název fronty delší než 48 znaků, bude jeho délka oříznuta na maximální délku 48 znaků, ale nebude vrácena žádná chybová zpráva.
- Před zadáním transakce stiskněte klávesu CLEAR.
- CSQ4CCJ1 může správně zobrazit pouze znakové zprávy, protože se zobrazí pouze do zobrazení prvního znaku NULL (\0).
- Do číselného pole zadejte libovolné číslo v rozsahu 1 až 9999. Hodnota, kterou zadáte, by měla být kladné číslo. Chcete-li například získat jedinou 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 jazyce C a 9 950 bajtů v jazyce COBOL jsou oříznuty. Důvodem je způsob použití front dočasného úložiště CICS .
- Pro oba programy, CSQ4CCK1 a CSQ4CVK1, zadejte do parametru get hodnotu B, chcete-li procházet zprávy, jinak zadejte hodnotu D. Provádí destruktivní volání MQGET. Pokud zadáte jinou hodnotu, obdržíte chybovou zprávu.
- Pro oba programy, CSQ4CCJ1 a CSQ4CVJ1, zadejte do parametru syncpoint hodnotu S pro načtení zpráv v synchronizačním bodu. Zadáte-li do parametru syncpoint hodnotu N, budou volání MQGET vydána mimo synchronizační bod. Pokud zadáte jinou hodnotu, obdržíte chybovou zprávu.

z/OS *Ukázka procházení na z/OS*

Ukázka Procházet je dávková aplikace, která demonstruje, jak procházet zprávy ve frontě pomocí volání MQGET.

Aplikace projde všemi zprávami ve frontě a vytiskne prvních 80 bajtů každé z nich. Tuto aplikaci můžete použít k vyhledání zpráv ve frontě bez jejich změny.

Zdrojové programy a ukázkový skript JCL jsou dodávány v jazycích COBOL, assembler, PL/I a C (viz Tabulka 173 na stránce 1131).

Chcete-li spustit aplikaci, upravte a spusťte ukázkový skript JCL, jak je popsáno v tématu “Příprava a spuštění ukázkových aplikací pro dávkové prostředí v systému z/OS” na stránce 1129. Můžete se podívat na zprávy v jedné ze svých vlastních front zadáním názvu fronty ve spouštěném JCL.

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 zprávu `Konec sestavy` . Dojde-li k chybě u některého z volání MQI, budou do výstupní datové sady přidány kódy dokončení a příčiny.

z/OS *Návrh ukázky procházení na webu z/OS*

Ukázková aplikace Browse používá jeden programový modul; jeden je poskytován v každém z podporovaných programovacích jazyků.

Tok logikou programu je:

1. Otevřete datovou sadu tisku a vytiskněte titulní řádek sestavy. Zkontrolujte, zda názvy správce front a fronty byly předány ze spuštění JCL. Pokud byly oba názvy předány, vytiskněte řádky sestavy, které obsahují názvy. Pokud ne, vytiskněte chybovou zprávu, zavřete datovou sadu tisku a zastavte zpracování.

   Způsob, jakým program testuje parametry předávané z JCL, závisí na jazyku, ve kterém je program napsán; další informace viz "Aspekty návrhu závislé na jazyku na webu z/OS" na stránce 1145.
2. Připojte se ke správci front pomocí volání MQCONN. Není-li toto volání úspěšné, vytiskněte kódy dokončení a příčiny, zavřete datovou sadu tisku a zastavte zpracování.
3. Otevřete frontu pomocí volání MQOPEN s volbou MQOO_BROWSE. Na vstupu tohoto volání program použije manipulátor připojení vrácený v kroku "2" na stránce 1144. Pro strukturu deskriptoru objektu (MQOD) používá výchozí hodnoty pro všechna pole kromě názvu fronty (který byl předán v kroku "1" na stránce 1144). Není-li toto volání úspěšné, vytiskněte kódy dokončení a příčiny, zavřete datovou sadu tisku a zastavte zpracování.
4. Procházejte první zprávu ve frontě pomocí volání MQGET. Při vstupu do tohoto volání program uvádí:
   - Obslužné rutiny připojení a fronty z kroků "2" na stránce 1144 a "3" na stránce 1144
   - Struktura MQMD se všemi poli nastavenými na počáteční hodnoty
   - Dvě možnosti:
     - MQGMO_BROWSE_FIRST
     - MQGMO_ACCEPT_TRUNCATED_MSG
   - Vyrovnávací paměť o velikosti 80 bajtů pro uchování dat zkopírovaných ze zprávy

   Volba MQGMO_ACCEPT_TRUNCATED_MSG umožňuje dokončení volání i v případě, že je zpráva delší než 80bajtová vyrovnávací paměť určená ve volání. Pokud je zpráva delší než vyrovnávací paměť, zpráva je oříznuta, aby se vešla do vyrovnávací paměti, a kódy dokončení a příčiny jsou nastaveny tak, aby to zobrazovaly. Ukázka byla navržena tak, aby byly zprávy zkráceny na 80 znaků, aby byla sestava snadno čitelná. Velikost vyrovnávací paměti je nastavena příkazem `DEFINE` , takže ji můžete snadno změnit, pokud chcete.
5. Proveďte následující smyčku, dokud volání MQGET neselže:
   a. Vytiskněte řádek sestavy zobrazující:
      - Pořadové číslo zprávy (počet operací procházení).
      - Skutečná délka zprávy (ne zkrácená délka). Tato hodnota je vrácena v poli `DataLength` volání MQGET.
      - Prvních 80 bajtů dat zprávy.

   b. Resetovat pole `MsqId` a `CorrelId` struktury MQMD na hodnoty null

   c. Procházejte další zprávu pomocí volání MQGET s těmito dvěma volbami:
      - MQGMO_BROWSE_NEXT
      - MQGMO_ACCEPT_TRUNCATED_MSG
6. Pokud se volání MQGET nezdaří, otestujte kód příčiny, abyste zjistili, zda se volání nezdařilo, protože se kurzor procházení dostal na konec fronty. V tomto případě vytiskněte zprávu `Konec sestavy` a přejděte na krok "7" na stránce 1144 ; jinak vytiskněte kódy dokončení a příčiny, zavřete datovou sadu tisku a zastavte zpracování.
7. Zavřete frontu pomocí volání MQCLOSE s popisovačem objektu vráceným v kroku "3" na stránce 1144.
8. Odpojte se od správce front pomocí volání MQDISC s manipulátoru připojení vráceným v kroku "2" na stránce 1144.

9. Zavřete datovou sadu tisku a zastavte zpracování.

z/OS *Aspekty návrhu závislé na jazyku na webu z/OS*

Zdrojové moduly jsou k dispozici pro ukázku Procházet ve čtyřech programovacích jazycích.

Mezi zdrojovými moduly existují dva hlavní rozdíly:

- Při testování parametrů předaných ze spuštění JCL moduly COBOL, PL/I a assembleru vyhledávají znak čárky (,). Pokud JCL projde `PARM=(,LOCALQ1)`, aplikace se pokusí otevřít frontu LOCALQ1 ve výchozím správci front. Pokud za čárkou (nebo bez čárky) není žádný název, aplikace vrátí chybu. Modul C nevyhledává znak čárky. Pokud JCL předá jeden parametr (například `PARM='LOCALQ1'`), modul C použije tento parametr jako název fronty ve výchozím správci front.
- Aby byl modul assembleru jednoduchý, používá při vytváření tiskové sestavy formát data *yy/ddd* (například 05/116). Ostatní moduly používají datum kalendáře ve formátu *mm/dd/rr* .

z/OS *Ukázka Tisk zprávy na webu z/OS*

Ukázka Tisk zprávy je dávková aplikace, která demonstruje, jak odebrat všechny zprávy z fronty pomocí volání MQGET.

Ukázka Tisk zprávy používá tři parametry:

1. Název správce front
2. Název zdrojové fronty
3. Volitelný parametr pro vlastnosti

Pro každou zprávu také vytiskne pole deskriptoru zprávy následovaná daty zprávy. Program vytiskne data jak hexadecimálně, tak jako znaky (jsou-li tisknutelná). Není-li znak tisknutelný, program jej nahradí tečkou (.). Program můžete použít při diagnostice problémů s aplikací, která vkládá zprávy do fronty.

Přípustné hodnoty pro parametr vlastnosti jsou:

Tabulka 185. Přípustné hodnoty pro parametr vlastnosti

Hodnota	Chování
0	Výchozí chování. Vlastnosti, které jsou doručeny aplikaci, závisí na atributu fronty **`PropertyControl`** , ze kterého je zpráva načtena.

1	Vytvoří se popisovač zprávy a použije se s MQGET. Vlastnosti zprávy s výjimkou těch, které jsou obsaženy v deskriptoru zprávy (nebo rozšíření), jsou zobrazeny podobným způsobem jako deskriptor zprávy. Příklad:
	`****Message properties****`
	`  property name: property value`
	Nebo pokud nejsou k dispozici žádné vlastnosti:
	`****Message properties****`
	`  None`
	Číselné hodnoty jsou zobrazeny pomocí `printf`, řetězcové hodnoty jsou uzavřeny v apostrofech a bajtové řetězce jsou uzavřeny do X a apostrofů, stejně jako pro deskriptor zprávy.

2	Je zadána volba MQGMO_NO_PROPERTIES, takže budou vráceny pouze vlastnosti deskriptoru zpráv.
3	Je zadána volba MQGMO_PROPERTIES_FORCE_MQRFH2 , aby byly v datech zprávy vráceny všechny vlastnosti.

Tabulka 185. Přípustné hodnoty pro parametr vlastnosti (pokračování)	
Hodnota	**Chování**
4	Je zadána volba MQGMO_PROPERTIES_COMPATIBILITY, aby bylo možné vrátit všechny vlastnosti v závislosti na tom, zda je vlastnost IBM MQ zahrnuta, jinak jsou vlastnosti zrušeny.

Aplikaci můžete změnit tak, aby procházala zprávy, a neodebírat ji z fronty. Chcete-li to provést, proveďte kompilaci s volbou -DBROWSE, abyste definovali makro BROWSE, jak je uvedeno v části “Návrh ukázky Tisk zprávy na webu z/OS” na stránce 1147. Spustitelný kód je uveden v knihovně SCSQLOAD. Modul CSQ4BCG0 je sestaven pomocí -DBROWSE; modul CSQ4BCG1 destruktivně čte frontu.

Aplikace má jeden zdrojový program, který je napsán v jazyce C. K dispozici je také ukázkový kód JCL spuštění (viz Tabulka 174 na stránce 1131).

Chcete-li spustit aplikaci, upravte a spusťte ukázkový skript JCL, jak je popsáno v tématu “Příprava a spuštění ukázkových aplikací pro dávkové prostředí v systému z/OS” na stránce 1129. Když spustíte aplikaci (a ve frontě jsou nějaké zprávy), bude výstupní datová sada vypadat jako v souboru Obrázek 139 na stránce 1147.

```
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 139. Příklad sestavy z ukázkové aplikace Tisk zprávy

z/OS *Návrh ukázky Tisk zprávy na webu z/OS*

Ukázková aplikace Print Message používá jediný program napsaný v jazyce C.

Tok logikou programu je:

1. Zkontrolujte, zda názvy správce front a fronty byly předány ze spuštění JCL. Pokud ne, 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; jinak vytiskněte název správce front.
3. Otevřete frontu pomocí volání MQOPEN s volbou MQOO_INPUT_SHARED.

   Poznámka: Pokud chcete, aby aplikace procházala zprávy a neodebírala je z fronty, zkompilujte ukázku pomocí volby -DBROWSE nebo přidejte `#define BROWSE` na začátek zdroje. Pokud tak učiníte, preprocesor maker přidá řádek v programu, který v kompilaci vybere volbu MQOO_BROWSE.

   Na vstupu tohoto volání program použije manipulátor připojení vrácený v kroku “2” na stránce 1147. Pro strukturu deskriptoru objektu (MQOD) používá výchozí hodnoty pro všechna pole kromě názvu

fronty (který byl předán v kroku “1” na stránce 1147). Není-li toto volání úspěšné, vytiskněte kódy dokončení a příčiny a zastavte zpracování; jinak vytiskněte název fronty.

4. Použijete-li popisovač zprávy k získání vlastností zprávy, použijte MQCRTMH k vytvoření takového popisovače pro použití s následnými voláními MQGET. Není-li toto volání úspěšné, vytiskněte kódy dokončení a příčiny a zastavte zpracování.
5. Nastavte volby získání zprávy tak, aby odrážely akci požadavku pro všechny vlastnosti zprávy.
6. Proveďte následující smyčku, dokud volání MQGET neselže:
   a. Inicializujte vyrovnávací paměť na mezery, aby data zprávy nebyla poškozena žádnými daty, která jsou již ve vyrovnávací paměti.
   b. Nastavte pole `MsgId` a `CorrelId` struktury MQMD na hodnotu null, aby volání MQGET vybrala první zprávu z fronty.
   c. Získejte zprávu z fronty pomocí volání MQGET. Při vstupu do tohoto volání program uvádí:
      - Obslužné rutiny připojení a objektu z kroků “2” na stránce 1147 a “3” na stránce 1147.
      - Struktura MQMD se všemi poli nastavenými na počáteční hodnoty. (`MsgId` a `CorrelId` jsou resetovány na hodnoty null pro každé volání MQGET.)
      - Volba MQGMO_NO_WAIT.

        Poznámka: Pokud chcete, aby aplikace procházala zprávy a neodebírala je z fronty, zkompilujte ukázku pomocí volby -DBROWSE nebo přidejte `#define BROWSE` na začátek zdroje. Pokud tak učiníte, preprocesor maker přidá do kompilace řádek v programu, který vybere volbu MQGMO_BROWSE_NEXT. Je-li tato volba použita ve volání pro frontu, pro kterou nebyl dříve použit žádný kurzor procházení s aktuálním popisovačem objektu, je kurzor procházení umístěn logicky před první zprávou.
      - Vyrovnávací paměť o velikosti 64KB pro uchování dat zkopírovaných ze zprávy.
   d. Zavolejte subrutinu printMD . Tímto se vytiskne název každého pole v deskriptoru zprávy následovaný jeho obsahem.
   e. Pokud jste v kroku “4” na stránce 1148 vytvořili popisovač zprávy, vyvolejte subrutinu printProperties a zobrazte vlastnosti zprávy.
   f. Vytiskněte délku zprávy následovanou daty zprávy. Každý řádek dat zprávy je v tomto formátu:
      - Relativní pozice (hexadecimální) této části dat
      - 16 bajtů hexadecimálních dat
      - Stejných 16 bajtů dat ve znakovém formátu, pokud jsou tisknutelná (netisknutelné znaky jsou nahrazeny tečkami)
7. Pokud se volání MQGET nezdaří, otestujte kód příčiny a zjistěte, zda se volání nezdařilo, protože ve frontě nejsou žádné další zprávy. V tomto případě vytiskněte zprávu: Žádné další zprávy; jinak vytiskněte kódy dokončení a příčiny. V obou případech přejděte na krok “9” na stránce 1148.

   Poznámka: Volání MQGET se nezdaří, pokud najde zprávu, která má více než 64KB dat. Chcete-li změnit program tak, aby zpracovoval větší zprávy, můžete provést jednu z následujících akcí:

   - Přidejte volbu MQGMO_ACCEPT_TRUNCATED_MSG do volání MQGET, aby volání získávala prvních 64KB dat a zbytek vyřadí.
   - Přinutit program opustit zprávu ve frontě, když ji najde s tímto množstvím dat
   - Zvětšit velikost vyrovnávací paměti
8. Pokud jste v kroku “4” na stránce 1148 vytvořili popisovač zprávy, vyvolejte MQDLTMH a odstraňte jej.
9. Zavřete frontu pomocí volání MQCLOSE s popisovačem objektu vráceným v kroku “3” na stránce 1147.
10. Odpojte se od správce front pomocí volání MQDISC s manipulátoru připojení vráceným v kroku “2” na stránce 1147.

z/OS *Ukázka Atributy fronty na systému z/OS*

Ukázka Atributy fronty je aplikace CICS v režimu konverzace, která demonstruje použití volání MQINQ a MQSET.

Ukazuje, jak zjistit hodnoty atributů front **InhibitPut** a **InhibitGet** a jak je změnit tak, aby programy nemohly vkládat zprávy do fronty nebo z ní získávat zprávy. Při testování programu můžete chtít *zamknout* frontu tímto způsobem.

Chcete-li zabránit náhodnému rušení vlastních front, tato ukázka pracuje pouze na objektu fronty, který má znaky CSQ4SAMP v prvních osmi bajtech svého názvu. Zdrojový kód však obsahuje komentáře, které vám ukážou, jak toto omezení odebrat.

Zdrojové programy jsou dodávány v jazycích COBOL, assembler a C (viz Tabulka 180 na stránce 1135).

Verze ukázky v jazyku asembleru používá znovu zadatelný kód. Za tímto účelem si všimnete, že kód pro každé volání MQI v dané 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 transakci dodanou nástrojem CICS Execution Diagnostic Facility (CEDF).) Další informace o psaní reenterovatelných programů naleznete v tématu Kódování v System/390 assembleru.

Chcete-li spustit aplikaci, spusťte systém CICS a použijte následující transakce CICS :

- Pro COBOL, MVC1
- V případě jazyka asembleru MAC1
- Pro C, MCC1

Název libovolné z těchto transakcí můžete změnit změnou datové sady CSD uvedené v kroku 3.

Návrh vzorku

Když spustíte ukázku, zobrazí mapu obrazovky, která obsahuje pole pro:

- Název fronty
- Požadavek uživatele (platné akce jsou: inquire, allow nebo inhibit)
- 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 zadáte do prvních dvou polí. Zkontroluje, zda název fronty začíná znaky `CSQ4SAMP` a zda jste do pole Akce zadali jeden ze tří platných požadavků. Aplikace převede veškerý vstup na velká písmena, takže nemůžete použít žádné fronty s názvy, které obsahují malá písmena.

Zadáte-li do pole **Akce** hodnotu `inquire` , bude tok logikou programu následující:

1. Otevřete frontu pomocí volání MQOPEN s volbou MQOO_INQUIRE.
2. Volání MQINQ pomocí selektorů MQIA_INHIBITORY-get a MQIA_INHIBITORY-put
3. Zavřít frontu pomocí volání MQCLOSE
4. Analyzujte atributy vrácené v parametru **IntAttrs** volání MQINQ a podle potřeby přesuňte slova INHIBITED nebo ALLOWED do příslušných polí obrazovky.

Zadáte-li do pole **Akce** hodnotu `inhibit` , bude tok logikou programu následující:

1. Otevřete frontu pomocí volání MQOPEN s volbou MQOO_SET.
2. Volejte příkaz MQSET s použitím selektorů MQIA_INHIB_GET a MQIA_INHIB_PUT a s hodnotami MQQA_GET_INHIBITED a MQQA_PUT_INHIBITED v parametru **IntAttrs** .
3. Zavřít frontu pomocí volání MQCLOSE

4. Přesunout slovo BLOKOVÁNO do příslušných polí obrazovky

Zadáte-li do pole **Akce** hodnotu `allow` , aplikace provede podobné zpracování jako pro blokovací požadavek. Jedinými rozdíly jsou nastavení atributů a slov zobrazených na obrazovce.

Když aplikace otevře frontu, použije výchozí manipulátor připojení ke správci front. (Produkt CICS vytvoří připojení ke správci front při spuštění systému CICS .) Aplikace může v této fázi zachycovat následující chyby:

- Aplikace není připojena ke správci front.
- Fronta neexistuje.
- Uživatel nemá oprávnění pro přístup k frontě.
- Aplikace nemá oprávnění k otevření fronty.

V případě jiných chyb MQI aplikace zobrazí kódy dokončení a příčiny.

z/OS *Ukázka Správce pošty na webu z/OS*

Ukázková aplikace Mail Manager je sada programů, která demonstruje odesílání a příjem zpráv, a to jak v rámci jednoho prostředí, tak v různých prostředích. Aplikace je jednoduchý elektronický poštovní systém, který umožňuje uživatelům vyměňovat si zprávy, i když používají různé správce front.

Aplikace demonstruje, jak vytvořit fronty pomocí volání MQOPEN a vložením příkazů 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 Správce pošty na z/OS*

Správce pošty 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 poštovním frontám a frontám přezdívek jak z TSO, tak z CICS , pokud jsou jejich přihlašovací ID uživatelů v každém systému stejná.

Před odesláním zpráv jinému správci front je nutné nastavit kanál zpráv pro tohoto správce front. K tomu použijte funkci řízení kanálu IBM MQ, popsanou v části Funkce řízení kanálu.

Příprava ukázky pro prostředí TSO

Postupujte takto:

1. Připravte vzorek podle popisu v části “Příprava ukázkových aplikací pro prostředí TSO na systému z/OS” na stránce 1132.
2. Upravte CLIST poskytnutý pro ukázku tak, aby definoval:
   - Umístění panelů
   - Umístění souboru zpráv
   - Umístění zaváděcích modulů
   - Název správce front, kterého chcete použít s aplikací

   Pro každou jazykovou verzi ukázky je k dispozici samostatný CLIST:
   - Pro verzi v jazyce COBOL: CSQ4RVD1
   - Pro verzi C: CSQ4RCD1
3. Zkontrolujte, zda jsou ve správci front k dispozici fronty používané aplikací. (Fronty jsou definovány v CSQ4CVD.)

Poznámka: VS COBOL II nepodporuje multitasking s ISPF. To znamená, že ukázkovou aplikaci Mail Manager nelze použít na obou stranách rozdělené obrazovky. Pokud tak učiníte, výsledky jsou nepředvídatelné.

z/OS *Spuštění ukázky Správce pošty v systému z/OS*

Chcete-li spustit ukázku v prostředí CICS Transaction Server for z/OS , spusťte transakci MAIL. Pokud jste se dosud nepřihlásili k produktu CICS, aplikace vás vyzve k zadání ID uživatele, kterému může odeslat vaši poštu.

Po spuštění aplikace se otevře fronta pošty. 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

v položkách CICS
Přihlášení uživatele CICS nebo ID uživatele zadané uživatelem při zobrazení výzvy při spuštění správce pošty

Všechny části názvů front, které správce pošty používá, musí být uvedeny 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 frontě pošty. Každá z voleb nabídky zobrazí další panel:

Číst příchozí poštu
Správce pošty zobrazí seznam zpráv, které jsou ve vaší frontě pošty. (Zobrazí se pouze prvních 99 zpráv ve frontě.) Příklad tohoto panelu viz Obrázek 142 na stránce 1154. Když vyberete zprávu z tohoto seznamu, zobrazí se její obsah (viz Obrázek 143 na stránce 1155).

Odeslat e-mail
Panel vás vyzve k zadání:

- Jméno uživatele, kterému chcete odeslat zprávu
- Název správce front, který vlastní svou frontu pošty
- Text vaší zprávy

Do pole Jméno uživatele můžete zadat buď ID uživatele, nebo přezdívku, kterou jste vytvořili pomocí správce pošty. Pole názvu správce front můžete ponechat prázdné, pokud je fronta pošty uživatele vlastněna stejným správcem front, kterého používáte, a musíte je ponechat prázdné, pokud jste zadali přezdívku do pole jména uživatele:

- Zadáte-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 název správce front, program odešle zprávu do fronty pošty, která je definována těmito dvěma názvy.

Chcete-li například odeslat zprávu uživateli JONESM ve vzdáleném správci front QM12, můžete jim zprávu odeslat jedním ze dvou způsobů:

- Pomocí obou polí určete uživatele JONESM ve správci front QM12.
- Definujte přezdívku (například MARY) pro tohoto uživatele a odešlete mu zprávu vložením MARY do pole jména uživatele a nic do pole názvu správce front.

Vytvořit přezdívku
Můžete definovat snadno zapamatovatelné jméno, které můžete použít při odesílání zprávy jinému uživateli, kterého často kontaktujete. Budete vyzváni k zadání ID uživatele jiného uživatele a názvu správce front, který vlastní svou frontu pošty.

Přezdívky jsou fronty, které mají názvy ve tvaru CSQ4SAMP.MAILMGR. *userid.nickname*, kde *ID_uživatele* je vaše vlastní ID uživatele a *přezdívka* je přezdívka, kterou chcete použít. S názvy strukturovanými tímto způsobem mohou mít uživatelé vlastní sadu přezdívek.

Typ fronty, kterou program vytvoří, závisí na tom, jak vyplníte pole na panelu Vytvořit přezdívku:

- Pokud zadáte pouze jméno uživatele nebo název správce front je stejný jako název správce front, ke kterému je připojen správce pošty, program vytvoří alias fronty.
- Zadáte-li jméno uživatele i název správce front (a správce front není ten, ke kterému je připojen správce pošty), program vytvoří lokální definici vzdálené fronty. Program nekontroluje existenci fronty, na kterou se tato definice interpretuje, ani to, že existuje vzdálený správce front.

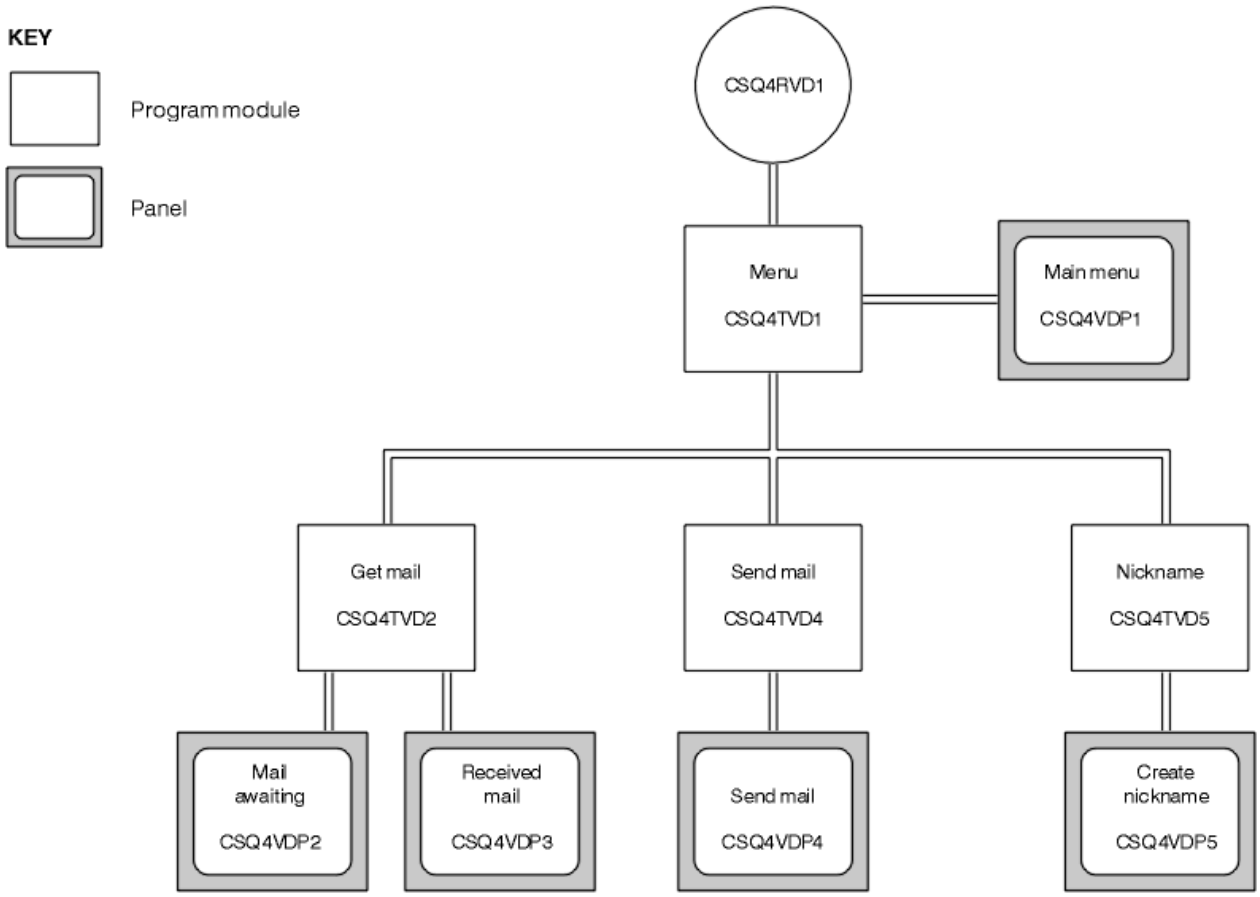
Pokud je například vaším vlastním ID uživatele SMITHK a vytvoříte přezdívku s názvem MARY pro uživatele JONESM (který používá vzdáleného správce front QM12), vytvoří program přezdívky lokální definici vzdálené fronty s názvem CSQ4SAMP.MAILMGR.SMITHK.MARY. Tato definice se interpretuje jako fronta pošty Marie, což je CSQ4SAMP.MAILMGR.JONESM ve správci front QM12. Pokud používáte správce front QM12 sami, program místo toho vytvoří alias fronty se stejným názvem (CSQ4SAMP.MAILMGR.SMITHK.MARY).

Verze C aplikace TSO lépe využívá schopnosti ISPFpro zpracování zpráv než verze jazyka COBOL. Můžete si všimnout, že verze jazyka C a COBOL zobrazují různé chybové zprávy.

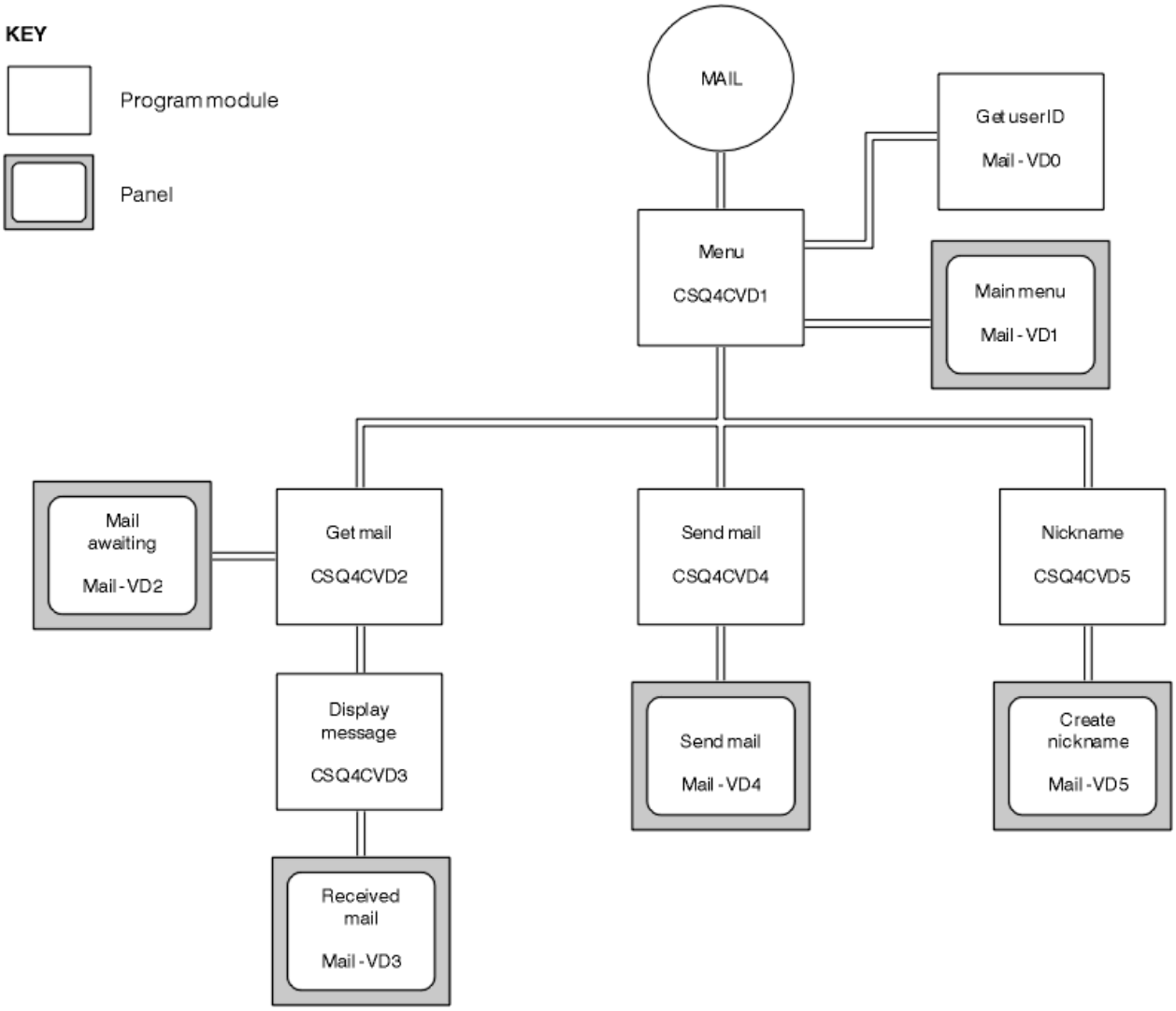
z/OS *Návrh ukázky Správce pošty na webu z/OS*

Následující sekce popisují všechny programy, které tvoří ukázkovou aplikaci Mail Manager.

Vztahy mezi programy a panely, které aplikace používá, jsou zobrazeny v souboru Obrázek 140 na stránce 1152 pro verzi TSO a v souboru Obrázek 141 na stránce 1153 pro verzi CICS Transaction Server for z/OS .



Obrázek 140. Programy a panely pro verze TSO správce pošty



Obrázek 141. Programy a panely pro verzi CICS správce pošty

z/OS *Program nabídky na z/OS*

V prostředí TSO je program nabídky vyvolán CLIST. V prostředí CICS je program vyvolán transakcí MAIL.

Program nabídky (CSQ4TVD1 pro TSO, CSQ4CVD1 pro CICS) je počáteční program v sadě. Zobrazí nabídku (CSQ4VDP1 pro TSO, VD1 pro CICS) a vyvolá ostatní programy, když jsou vybrány z nabídky.

Program nejprve získá ID uživatele:

- Ve verzi programu CICS , pokud se uživatel přihlásil k produktu CICS, je ID uživatele získáno pomocí příkazu CICS ASSIGN USERID. Pokud se uživatel nepřihlásil, program zobrazí přihlašovací panel (CSQ4VD0), který vyzve uživatele k zadání ID uživatele. V rámci tohoto programu neexistuje žádné zpracování zabezpečení; uživatel může zadat libovolné ID uživatele.
- Ve verzi TSO je ID uživatele získáno z TSO v CLIST. Předává se do programu nabídky jako proměnná ve sdíleném fondu ISPF .

Poté, co program získá ID uživatele, zkontroluje, zda má uživatel frontu pošty (CSQ4SAMP.MAILMGR. *userid*). Pokud fronta pošty neexistuje, program ji vytvoří vložením zprávy do vstupní fronty systémového příkazu. Zpráva obsahuje příkaz IBM MQ for z/OS DEFINE QLOCAL. Definice objektu, kterou tento příkaz používá, nastaví maximální hloubku fronty na 9999 zpráv.

Program také vytvoří dočasnou dynamickou frontu pro zpracování odpovědí ze vstupní fronty systémového příkazu. K tomu program používá volání MQOPEN s uvedením SYSTEM.DEFAULT.MODEL.QUEUE jako šablonu pro dynamickou frontu. Správce front vytvoří dočasnou

dynamickou frontu s názvem, který má předponu CSQ4SAMP; . Zbytek názvu je generován správcem front.

Program poté otevře frontu pošty uživatele a vyhledá počet zpráv ve frontě dotazem na aktuální hloubku fronty. K tomu program používá volání MQINQ a určuje selektor MQIA_CURRENT_Q_DEPTH.

Program pak provede smyčku, která zobrazí nabídku a zpracuje výběr, který uživatel provede. Smyčka se zastaví, když uživatel stiskne klávesu PF3 . Když je proveden platný výběr, spustí se příslušný program; jinak se zobrazí chybová zpráva.

z/OS *Získat-poštu a zobrazit-programy zpráv na z/OS*

Ve verzích TSO aplikace jsou funkce get-mail a display-message prováděny stejným programem (CSQ4TVD2). Ve verzi CICS aplikace jsou tyto funkce prováděny samostatnými programy (CSQ4CVD2 a CSQ4CVD3).

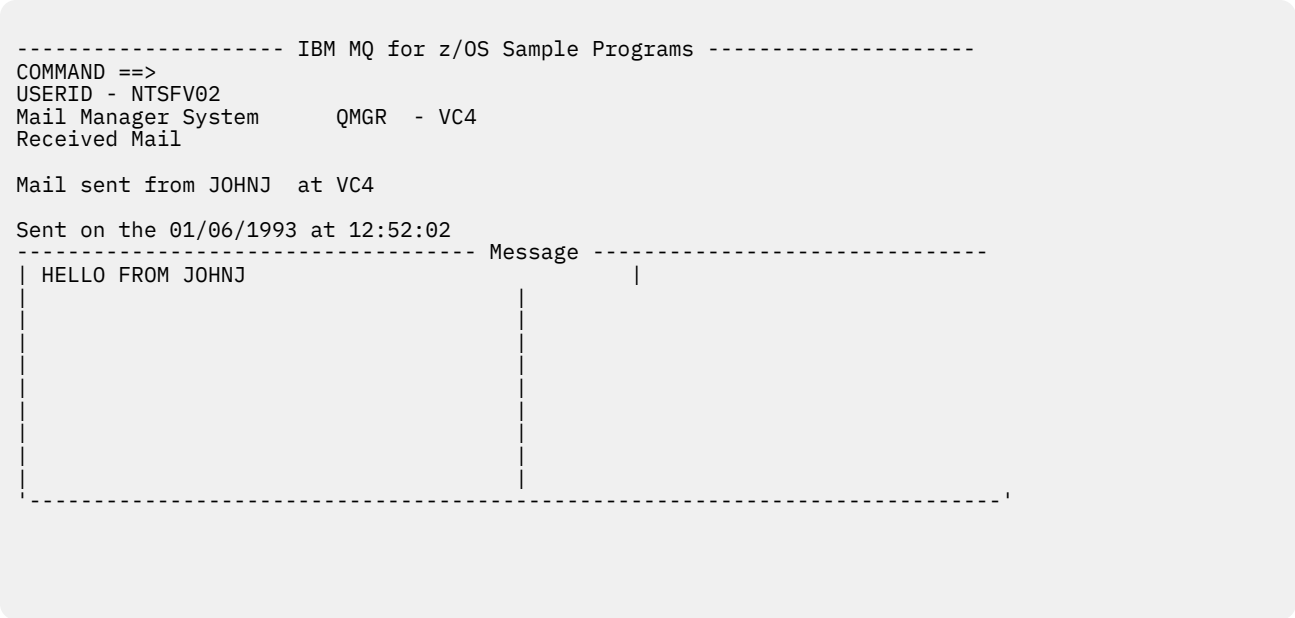
panel Čekání na poštu (CSQ4VDP2 pro TSO, VD2 pro CICS ; Viz Obrázek 142 na stránce 1154 , kde je uveden příklad) zobrazuje všechny zprávy, které jsou ve frontě pošty uživatele. K vytvoření tohoto seznamu používá program volání MQGET k procházení všech zpráv ve frontě a k ukládání informací o každé z nich. Kromě zobrazených informací program zaznamenává `MsgId` a `CorrelId` každé zprávy.

```
--------------------- 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 142. Příklad panelu zobrazujícího seznam čekajících zpráv

Na panelu Čekání na poštu může uživatel vybrat jednu zprávu a zobrazit obsah zprávy (viz příklad Obrázek 143 na stránce 1155). Program používá volání MQGET k odebrání této zprávy z fronty pomocí `MsgId` a `CorrelId` , které program zaznamenal při procházení všech zpráv. Toto volání MQGET se provádí pomocí volby MQGMO_SYNCPOINT. Program zobrazí obsah zprávy a poté deklaruje synchronizační bod: tímto se potvrdí volání MQGET, takže zpráva již neexistuje.



Obrázek 143. Příklad panelu zobrazujícího obsah zprávy

Zjevné rozšíření funkce poskytované správcem pošty je poskytnout uživateli možnost zanechat zprávu ve frontě po zobrazení jejího obsahu. Chcete-li to provést, musíte po zobrazení zprávy vrátit zpět volání MQGET, které zprávu odebere z fronty.

z/OS *Program pro odesílání pošty na z/OS*

Když uživatel dokončil panel Odeslat poštu (CSQ4VDP4 pro TSO, VD4 pro CICS), program pro odesílání pošty (CSQ4TVD4 pro TSO, CSQ4CVD4 pro CICS). vloží zprávu do fronty pošty příjemce.

K tomu 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 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.
- Pokud uživatel zadal jméno uživatele i název správce front, program odešle zprávu do fronty pošty, která je definována těmito dvěma názvy.

Program nepřijímá prázdné zprávy a odstraňuje úvodní mezery z každého řádku textu zprávy.

Pokud je volání MQPUT1 úspěšné, program zobrazí zprávu, která uvádí jméno uživatele a název správce front, do kterého byla zpráva vložena. Pokud je volání neúspěšné, program zkontroluje specificky kódy příčiny, které označují, že fronta nebo správce front neexistují; jedná se o MQRC_UNKNOWN_OBJECT_NAME a MQRC_UNKNOWN_OBJECT_Q_MGR. Program zobrazí vlastní chybovou zprávu pro každou z těchto chyb; pro ostatní chyby program zobrazí kódy dokončení a příčiny vrácené voláním.

z/OS *Program přezdívek na z/OS*

Když uživatel definuje přezdívku, program přezdívek (CSQ4TVD5 pro TSO, CSQ4CVD5 pro CICS) vytvoří frontu, která má přezdívku jako součást svého názvu.

Program to provede vložením zprávy do vstupní fronty systémového příkazu. Zpráva obsahuje příkaz IBM MQ for z/OS DEFINE QALIAS nebo DEFINE QREMOTE. Typ fronty, kterou program vytvoří, závisí na tom, jak uživatel vyplnil pole panelu Vytvořit přezdívku (CSQ4VDP5 pro TSO, VD5 pro CICS):

- Pokud uživatel zadal pouze jméno uživatele nebo název správce front je stejný jako název správce front, ke kterému je připojen správce pošty, program vytvoří alias fronty.

- Pokud uživatel zadal jméno uživatele i název správce front (a správce front není ten, ke kterému je připojen správce pošty), program vytvoří lokální definici vzdálené fronty. Program nekontroluje existenci fronty, na kterou se tato definice interpretuje, ani to, že existuje vzdálený správce front.

Program také vytvoří dočasnou dynamickou frontu pro zpracování odpovědí ze vstupní fronty systémového příkazu.

Pokud správce front nemůže vytvořit frontu přezdívek z důvodu, který program očekává (například fronta již existuje), zobrazí program vlastní chybovou zprávu. Pokud správce front nemůže vytvořit frontu z důvodu, který program neočekával, zobrazí se v programu až dvě chybové zprávy vrácené příkazovým serverem programu.

Poznámka: Pro každou přezdívku vytvoří program přezdívek pouze alias frontu nebo lokální definici vzdálené fronty. Lokální fronty, na které se tyto názvy front interpretují, jsou vytvořeny pouze v případě, že ID uživatele obsažené v přezdívce je použito ke spuštění aplikace Mail Manager.

z/OS *Ukázka kontroly kreditu na z/OS*

Ukázková aplikace Credit Check je sada programů, která demonstruje, jak používat mnoho funkcí poskytovaných produktem IBM MQ for z/OS. Ukazuje, jak mnoho programů komponenty aplikace může předávat zprávy mezi sebou pomocí technik řazení zpráv do fronty.

Ukázku lze spustit jako samostatnou aplikaci CICS . Chcete-li však předvést, jak navrhnout aplikaci pro řazení zpráv do fronty, která používá prostředky poskytované prostředím CICS i IMS , je jeden modul také dodáván jako program pro dávkové zpracování zpráv IMS . Toto rozšíření ukázky je popsáno v tématu “Rozšíření IMS k ukázce kontroly kreditu na webu z/OS” na stránce 1165.

Můžete také spustit ukázku na více než jednom správci front a odesílat zprávy mezi jednotlivými instancemi aplikace. Chcete-li tak učinit, viz “Ukázka kontroly kreditu s více správci front v systému z/OS” na stránce 1165.

Programy CICS jsou dodávány v jazycích C a COBOL. Jediný program IMS je dodáván pouze v jazyce C. Dodané datové sady jsou zobrazeny v Tabulka 182 na stránce 1136 a Tabulka 184 na stránce 1138.

Aplikace demonstruje metodu hodnocení rizika, když bankovní zákazníci žádají o půjčky. Aplikace ukazuje, jak by banka mohla pracovat dvěma způsoby při zpracování žádostí o úvěr:

- Při přímém jednání se zákazníkem chtějí zaměstnanci banky mít okamžitý přístup k informacím o účtu a úvěrovém riziku.
- Při vyřizování písemných žádostí mohou zaměstnanci banky podat řadu žádostí o informace o účtu a úvěrovém riziku a odpovědi řešit později.

Podrobnosti o finančních a bezpečnostních údajích v aplikaci jsou jednoduché, takže techniky řazení zpráv do fronty jsou jasné.

z/OS *Příprava a spuštění ukázky kontroly kreditu na webu z/OS*

Chcete-li připravit a spustit ukázku kontroly kreditu, postupujte takto:

1. Vytvořte datovou sadu VSAM, která obsahuje informace o některých vzorových účtech. Toto proveďte úpravou a spuštěním JCL dodaného v datové sadě CSQ4FILE.
2. Proveďte kroky uvedené v tématu “Příprava ukázkových aplikací pro prostředí CICS na systému z/OS” na stránce 1134. (Další kroky, které musíte provést, pokud chcete použít rozšíření IMS k ukázce, jsou popsány v části “Rozšíření IMS k ukázce kontroly kreditu na webu z/OS” na stránce 1165.)
3. Spustit monitor spouštěčů CKTI (dodávaný s produktem IBM MQ for z/OS) proti frontě CSQ4SAMP.INITIATION.QUEUE, pomocí CKQC transakce CICS .
4. Chcete-li spustit aplikaci, spusťte systém CICS a použijte transakci MVB1.
5. Na prvním panelu vyberte volbu **Okamžitý** nebo **Dávkový** dotaz.

   Panely okamžitých a dávkových dotazů jsou podobné; Obrázek 144 na stránce 1157 zobrazuje panel okamžitých dotazů.

```
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 144. Panel okamžitých dotazů pro ukázkovou aplikaci Credit Check

6. Do příslušných polí zadejte číslo účtu a částku půjčky. Pokyny k zadání informací do těchto polí naleznete v části “Zadávání informací do dotazovacích panelů” na stránce 1157 .

Zadávání informací do dotazovacích panelů

Ukázková aplikace Kontrola kreditu zkontroluje, zda data, která zadáte do pole **Požadovaná částka** na panelech dotazů, jsou ve formě celých čísel.

Zadáte-li jedno z následujících čísel účtů, aplikace v datové sadě VSAM CSQ4BAQ:

- 2222222222
- 3111234329
- 3256478962
- 3333333333
- 3501676212
- 3696879656
- 4444444444
- 5555555555
- 6666666666
- 7777777777

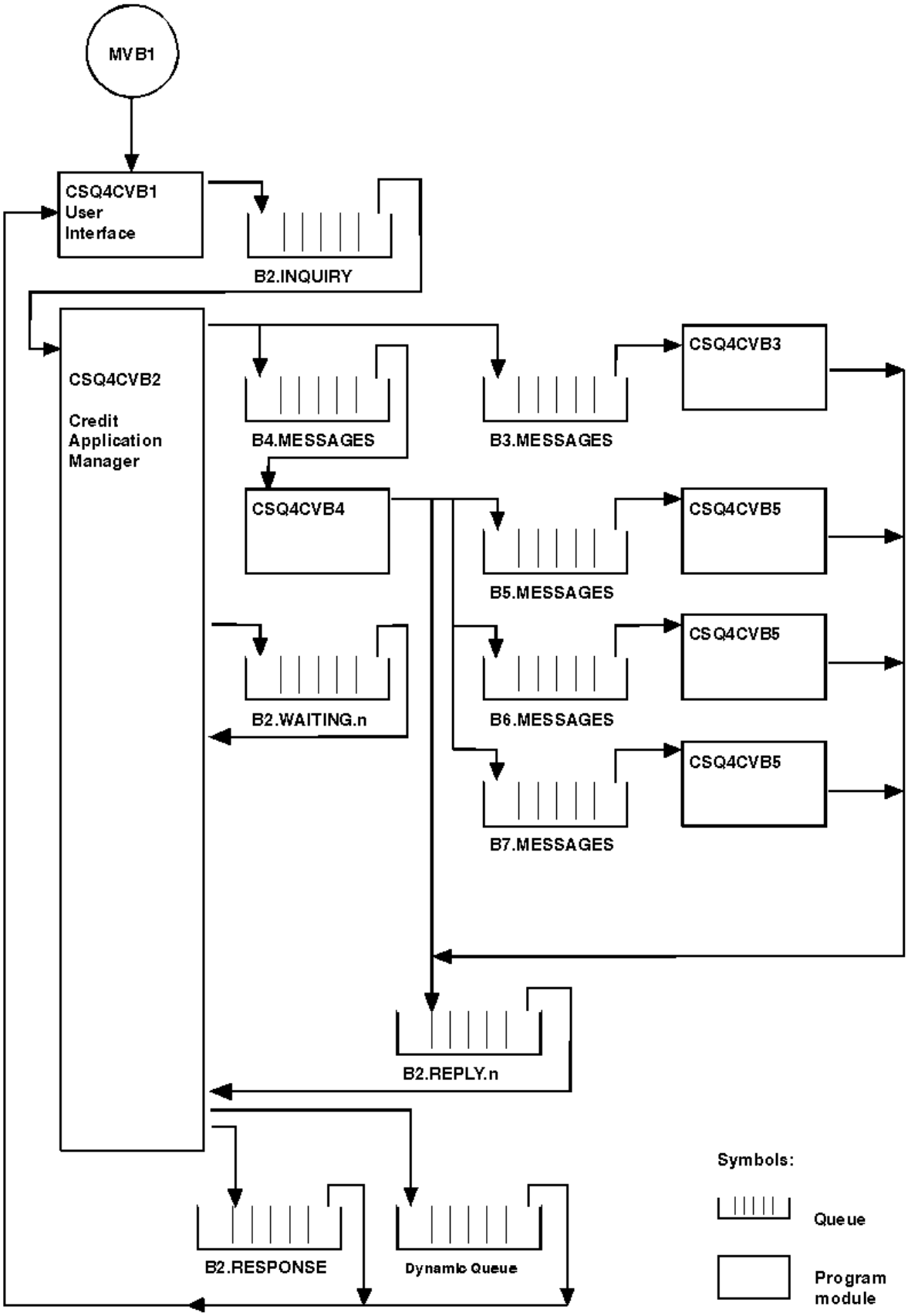
Do ostatních polí můžete zadat jakékoli nebo žádné informace. Aplikace zachová všechny informace, které zadáte, a vrátí stejné informace v sestavách, které vygeneruje.

z/OS *Návrh vzorku kontroly kreditu na z/OS*

Tato část popisuje návrh každého z programů, které tvoří ukázkovou aplikaci Credit Check.

Další informace o některých technikách, které byly zvažovány během návrhu aplikace, viz “Aspekty návrhu pro ukázku kontroly kreditu na webu z/OS” na stránce 1163.

Obrázek 145 na stránce 1158 zobrazuje programy, které tvoří aplikaci, a také fronty, které tyto programy obsluhují. Na tomto obrázku byla předpona CSQ4SAMP vynechána ze všech názvů front, aby byl obrázek srozumitelnější.



Obrázek 145. Programy a fronty pro ukázkovou aplikaci Credit Check (pouze programy v jazyce COBOL)

z/OS *Program uživatelského rozhraní (CSQ4CVB1) na systému z/OS*

Když spustíte transakci CICS transaction MVB1v dialogovém režimu, spustí se program uživatelského rozhraní pro aplikaci.

Tento program vkládá dotazové zprávy do fronty CSQ4SAMP.B2.INQUIRY a získává odpovědi na tyto dotazy z fronty pro odpovědi, kterou uvádí, když dotaz provádí. Z uživatelského rozhraní můžete zadat buď okamžité, nebo dávkové dotazy:

- V případě okamžitých dotazů program vytvoří dočasnou dynamickou frontu, kterou používá jako frontu pro odpovědi. To znamená, že každý dotaz má svou vlastní frontu pro odpověď.
- V případě dávkových dotazů získává program uživatelského rozhraní odpovědi z fronty CSQ4SAMP.B2.RESPONSE. Pro jednoduchost program získává odpovědi na všechny své dotazy z této jedné fronty pro odpovědi. Je snadné vidět, že banka může chtít použít oddělenou frontu odpovědí pro každého uživatele produktu MVB1, aby každý z nich viděl odpovědi pouze na ty dotazy, které zahájil.

Důležité rozdíly mezi vlastnostmi zpráv používaných v aplikaci v dávkovém a okamžitém režimu jsou:

- Pro dávkové zpracování mají zprávy nízkou prioritu, takže jsou zpracovány po všech žádostech o úvěr, které jsou zadány v okamžitém režimu. Zprávy jsou také trvalé, takže jsou obnoveny v případě, že je třeba restartovat aplikaci nebo správce front.
- Pro okamžitou práci mají zprávy vysokou prioritu, takže jsou zpracovány před jakýmikoli žádostmi o úvěr, které jsou zadány v dávkovém režimu. Zprávy také nejsou trvalé, takže jsou vyřazeny, pokud je třeba restartovat aplikaci nebo správce front.

Ve všech případech jsou však vlastnosti zpráv žádostí o úvěr šířeny v rámci celé aplikace. Takže například všechny zprávy, které jsou výsledkem požadavku s vysokou prioritou, budou mít také vysokou prioritu.

z/OS *Správce kreditních aplikací (CSQ4CVB2) na z/OS*

Program CAM (Credit Application Manager) provádí většinu zpracování pro aplikaci Credit Check.

Modul CAM je spuštěn monitorem spouštěče CKTI (dodávaným s produktem IBM MQ for z/OS), když dojde k události spouštěče v jedné z front CSQ4SAMP.B2.INQUIRY nebo fronta CSQ4SAMP.B2.REPLY. *n*, kde *n* je celé číslo, které identifikuje jednu ze sady front odpovědí. Zpráva spouštěče obsahuje data, která zahrnují název fronty, ve které došlo k události spouštěče.

CAM používá fronty s názvy ve tvaru CSQ4SAMP.B2.WAITING.n ukládá informace o dotazech, které zpracovává. Fronty jsou pojmenovány tak, že jsou všechny spárovány s frontou pro odpověď; například s frontou CSQ4SAMP.B2.WAITING.3 obsahuje vstupní data pro konkrétní dotaz a frontu CSQ4SAMP.B2.REPLY.3 obsahuje sadu zpráv s odpovědí (z programů, které se dotazují databází), které se vztahují ke stejnému dotazu. Chcete-li porozumět důvodům tohoto návrhu, viz “Samostatné fronty dotazů a odpovědí v CAM” na stránce 1163.

Logika spuštění

Pokud se událost spouštěče vyskytne ve frontě CSQ4SAMP.B2.INQUIRY, CAM otevře frontu pro sdílený přístup. Poté se pokusí otevřít každou frontu odpovědí, dokud nebude nalezena volná fronta. Pokud nemůže najít volnou frontu odpovědí, CAM zaprotokoluje skutečnost a ukončí se normálně.

Pokud se událost spouštěče vyskytne ve frontě CSQ4SAMP.B2.REPLY.n, CAM otevře frontu pro výlučný přístup. Pokud návratový kód hlásí, že objekt je již používán, CAM se ukončí normálně. Dojde-li k jakékoli jiné chybě, CAM zaprotokoluje chybu a ukončí se. CAM otevře odpovídající čekající frontu a dotazovou frontu, pak zahájí získávání a zpracování zpráv. Z čekající fronty CAM obnoví podrobnosti částečně dokončených dotazů.

V zájmu jednoduchosti v této ukázce jsou názvy použitých front uchovávány v programu. V obchodním prostředí budou názvy front pravděpodobně uloženy v souboru, ke kterému má program přístup.

Získání zprávy z fronty dotazů

CAM se nejprve pokusí získat zprávu z fronty dotazů pomocí volání MQGET s volbou MQGMO_SET_SIGNAL. Je-li zpráva k dispozici okamžitě, zpráva se zpracuje; není-li k dispozici žádná zpráva, nastaví se signál.

CAM se poté pokusí získat zprávu z fronty odpovědí znovu pomocí volání MQGET se stejnou volbou. Je-li zpráva k dispozici okamžitě, zpráva se zpracuje; jinak se nastaví signál.

Jsou-li oba signály nastaveny, program čeká na vyslání jednoho ze signálů. Pokud je vyslán signál, který označuje, že je zpráva k dispozici, je zpráva načtena a zpracována. Dojde-li k vypršení platnosti signálu nebo k ukončení činnosti správce front, program bude ukončen.

Zpracování zprávy načtené CAM

Zpráva načtená produktem CAM může být jednoho ze čtyř typů:

- Dotazová zpráva
- Zpráva odpovědi
- Zpráva šíření
- Neočekávaná nebo nechtěná zpráva

CAM zpracovává tyto zprávy, jak je popsáno v tématu “Zpracování zprávy načtené CAM na z/OS” na stránce 1160.

Odeslání odpovědi

Když CAM obdrží všechny odpovědi, které očekává pro dotaz, zpracuje odpovědi a vytvoří jedinou zprávu odpovědi. Konsoliduje do jedné zprávy všechna data ze všech zpráv odpovědi, které mají stejný `CorrelId`. Tato odpověď je vložena do fronty pro odpověď uvedené v původním požadavku na úvěr. Zpráva odpovědi je vložena do stejné pracovní jednotky, která obsahuje načtení konečné zprávy odpovědi. To má zjednodušit obnovu tím, že se ujistíte, že ve frontě CSQ4SAMP.B2.WAITING.n.

Navrácení částečně ukončených šetření

CAM kopíruje do fronty CSQ4SAMP.B2.WAITING.n všechny zprávy, které přijímá. Nastavuje pole deskriptoru zprávy takto:

- *`Priority`* je určeno typem zprávy:
  - Pro zprávy požadavku, priorita = 3
  - Pro datagramy, priorita = 2
  - Pro zprávy s odpovědí, priorita = 1
- Parametr *`CorrelId`* je nastaven na hodnotu *`MsgId`* zprávy požadavku na úvěr.
- Další pole MQMD jsou zkopírována z polí přijaté zprávy.

Po dokončení dotazu se zprávy pro určitý dotaz odeberou z čekající fronty během zpracování odpovědi. Proto čekající fronta kdykoli obsahuje všechny zprávy související s probíhajícími dotazy. Tyto zprávy se používají k obnovení podrobností probíhajících dotazů, pokud má být program restartován. Různé priority jsou nastaveny tak, aby dotazové zprávy byly obnoveny před šířením nebo zprávami odpovědí.

z/OS *Zpracování zprávy načtené CAM na z/OS*

Zpráva načtená správcem CAM (Credit Application Manager) může být jedním ze čtyř typů. Způsob, jakým CAM zpracovává zprávu, závisí na jejím typu.

Zpráva načtená produktem CAM může být jednoho ze čtyř typů:

- Dotazová zpráva
- Zpráva odpovědi

- Zpráva šíření
- Neočekávaná nebo nechtěná zpráva

CAM zpracovává tyto zprávy takto:

Dotazová zpráva
Dotazové zprávy pocházejí z programu uživatelského rozhraní. Vytvoří dotazovou zprávu pro každý požadavek na půjčku.

Pro všechny žádosti o úvěr vyžaduje CAM průměrný zůstatek kontrolního účtu zákazníka. Provádí to vložením zprávy požadavku do alias fronty CSQ4SAMP.B2.OUTPUT.ALIAS. Tento název fronty se interpretuje jako fronta CSQ4SAMP.B3.MESSAGES, který je zpracován programem kontrolního účtu CSQ4CVB3. Když CAM vloží zprávu do této alias fronty, určí odpovídající CSQ4SAMP.B2.REPLY.n fronta pro odpověď na frontu. Zde se používá alias fronta, takže program CSQ4CVB3 lze snadno nahradit jiným programem, který zpracovává základní frontu s jiným názvem. Chcete-li tak učinit, předefinujte alias frontu tak, aby se její název interpretoval na novou frontu. Také můžete přiřadit různá přístupová oprávnění k alias frontě a k základní frontě.

Pokud uživatel požádá o půjčku, která je větší než 10000 jednotek, CAM zahájí kontroly i v jiných databázích. Provádí to vložením zprávy požadavku do fronty CSQ4SAMP.B4.MESSAGES, které jsou zpracovány distribučním programem CSQ4CVB4. Proces obsluhující tuto frontu šíří zprávu do front obsluhovaných programy, které mají přístup k jiným záznamům, jako je historie kreditních karet, spořicí účty a hypoteční splátky. Data z těchto programů se vrátí do fronty pro odpověď uvedené v operaci vložení. Kromě toho tento program odešle zprávu šíření do fronty pro odpověď, aby určil, kolik zpráv šíření bylo odesláno.

V obchodním prostředí by distribuční program pravděpodobně přeformátoval poskytnutá data tak, aby odpovídala formátu požadovanému každým z ostatních typů bankovních účtů.

Jakákoli z odkazovaných front může být na vzdáleném systému.

Pro každou dotazovou zprávu iniciuje CAM záznam v tabulce záznamů dotazů (IRT) rezidentní v paměti. Tento záznam obsahuje:

- `MsgId` dotazové zprávy
- V poli ReplyExp se jedná o počet očekávaných odpovědí (rovnající se počtu odeslaných zpráv).
- V poli ReplyRec počet přijatých odpovědí (nula v této fázi).
- V poli PropsOut označuje, zda je očekávána zpráva šíření.

CAM zkopíruje dotazovou zprávu do čekající fronty s:

- `Priority` nastavit na hodnotu 3
- `CorrelId` nastaveno na `MsgId` dotazové zprávy
- Ostatní pole deskriptoru zpráv jsou nastavena na pole dotazové zprávy.

Zpráva šíření
Zpráva šíření obsahuje počet front, do kterých distribuční program postoupil dotaz. Zpráva je zpracována následujícím způsobem:

1. Do pole ReplyExp příslušného záznamu v projektu IRT přidejte počet odeslaných zpráv. Tato informace je ve zprávě.
2. Zvyšte o 1 pole ReplyRec záznamu v projektu IRT.
3. Snížení o 1 pole PropsOut záznamu v projektu IRT.
4. Zkopírujte zprávu do čekající fronty. CAM nastaví `Priority` na 2 a ostatní pole deskriptoru zprávy na ta, která jsou uvedena ve zprávě šíření.

Odpověď na zprávu
Zpráva odpovědi obsahuje odpověď na jeden z požadavků na program kontrolního účtu nebo na jeden z programů agenturního dotazu. Zprávy odpovědí jsou zpracovány následujícím způsobem:

1. Zvyšte o 1 pole ReplyRec záznamu v projektu IRT.

2. Zkopírujte zprávu do čekající fronty s hodnotou `Priority` nastavenou na 1 a ostatní pole deskriptoru zprávy nastavená na ta, která jsou ve zprávě odpovědi.
3. Pokud je hodnota ReplyRec = ReplyExpa hodnota PropsOut = 0, nastavte příznak MsgComplete .

Další zprávy

Aplikace neočekává jiné zprávy. Aplikace však může přijímat zprávy vysílané systémem nebo odpovídat na zprávy s neznámým `CorrelIds`.

CAM vkládá tyto zprávy do fronty CSQ4SAMP.DEAD.QUEUE, kde je lze zkontrolovat. Pokud tato operace vložení selže, zpráva se ztratí a program bude pokračovat. Další informace o návrhu této části programu viz "Jak ukázka zpracovává neočekávané zprávy" na stránce 1164.

z/OS *Program pro kontrolu účtu (CSQ4CVB3) na z/OS*

Program kontrolního účtu je spuštěn událostí spouštěče ve frontě CSQ4SAMP.B3.MESSAGES. Po otevření fronty tento program získá zprávu z fronty pomocí volání MQGET s volbou wait a s intervalem čekání nastaveným na 30 sekund.

Program hledá číslo účtu ve zprávě požadavku na úvěr v datové sadě VSAM CSQ4BAQ . Načte odpovídající název účtu, průměrný zůstatek a index úvěruschopnosti, nebo si poznamená, že číslo účtu není v datové sadě.

Program poté vloží zprávu odpovědi (pomocí volání MQPUT1) do fronty pro odpověď uvedené ve zprávě požadavku na úvěr. Pro tuto zprávu odpovědi program:

- Zkopíruje `CorrelId` zprávy žádosti o úvěr
- Používá volbu MQPMO_PASS_IDENTITY_CONTEXT

Program pokračuje v získávání zpráv z fronty, dokud nevyprší interval čekání.

z/OS *Distribuční program (CSQ4CVB4) na z/OS*

Distribuční program je spuštěn událostí spouštěče ve frontě CSQ4SAMP.B4.MESSAGES.

Chcete-li simulovat distribuci žádosti o úvěr jiným agenturám, které mají přístup k záznamům, jako je historie kreditních karet, spořicí účty a hypoteční splátky, program vloží kopii stejné zprávy do všech front v seznamu názvů CSQ4SAMP.B4.NAMELIST. Existují tři z těchto front s názvy ve tvaru CSQ4SAMP.B *n*.ZPRÁVY, kde *n* je 5, 6 nebo 7. V obchodní aplikaci mohou být agentury v oddělených umístěních, takže tyto fronty mohou být vzdálenými frontami. Chcete-li upravit ukázkovou aplikaci tak, aby to zobrazovala, prohlédněte si téma "Ukázka kontroly kreditu s více správci front v systému z/OS" na stránce 1165.

Distribuční program provádí následující kroky:

1. Ze seznamu názvů získá názvy front, které má program používat. Program to provede pomocí volání MQINQ s dotazem na atributy objektu seznamu názvů.
2. Otevře tyto fronty a také CSQ4SAMP.B4.MESSAGES.
3. Provede následující smyčku, dokud ve frontě 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 příslušného CSQ4SAMP.B2.REPLY.n fronta pro odpověď na frontu. Program zkopíruje soubor *`CorrelId`* zprávy žádosti o úvěr do těchto kopírovaných zpráv a použije volbu MQPMO_PASS_IDENTITY_CONTEXT ve volání MQPUT.
   c. Odešlete zprávu datagramu do fronty CSQ4SAMP.B2.REPLY.n ukazuje, kolik zpráv úspěšně vložil.
   d. Deklarujte synchronizační bod.

z/OS *Program pro dotazy agentur (CSQ4CVB5/CSQ4CCB5) na z/OS*

Program agency-query je dodáván jak jako program v jazyce COBOL, tak jako program v jazyce C. Oba programy mají stejný design. To ukazuje, že programy různých typů mohou snadno existovat v rámci aplikace IBM MQ a že programové moduly, které tvoří takovou aplikaci, mohou být snadno nahrazeny.

Instance programu je spuštěna událostí spouštěče v libovolné z těchto front:

- Pro program v jazyce COBOL (CSQ4CVB5):
  - CSQ4SAMP.B5.MESSAGES
  - CSQ4SAMP.B6.MESSAGES
  - CSQ4SAMP.B7.MESSAGES
- Pro program v jazyce C (CSQ4CCB5) se jedná o frontu CSQ4SAMP.B8.MESSAGES

Poznámka: Chcete-li použít program v jazyce C, musíte změnit definici seznamu názvů CSQ4SAMP.B4.NAMELIST nahradí frontu CSQ4SAMP.B7.MESSAGES s produktem CSQ4SAMP.B8.MESSAGES. Chcete-li to provést, můžete použít některou z následujících možností:

- Operační a ovládací panely IBM MQ for z/OS
- Příkaz ALTER NAMELIST
- Obslužný program CSQUTIL

Po otevření příslušné fronty tento program obdrží 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 žádosti o úvěr. Poté vytvoří odpověď, která obsahuje název fronty, kterou obsluhuje, a index úvěruschopnosti. Pro zjednodušení zpracování je index úvěruschopnosti vybrán náhodně.

Při vkládání zprávy odpovědi program používá volání MQPUT1 a:

- Zkopíruje `CorrelId` zprávy žádosti o úvěr
- Používá volbu MQPMO_PASS_IDENTITY_CONTEXT

Program odešle zprávu odpovědi do fronty pro odpověď uvedené ve zprávě žádosti o úvěr. (Název správce front, který vlastní frontu pro odpověď, je uveden také ve zprávě požadavku na úvěr.)

z/OS *Aspekty návrhu pro ukázku kontroly kreditu na webu z/OS*

Aspekty návrhu pro ukázku kontroly kreditu.

Toto téma obsahuje informace o:

- “Samostatné fronty dotazů a odpovědí v CAM” na stránce 1163
- “Jak ukázka zpracovává chyby” na stránce 1164
- “Jak ukázka zpracovává neočekávané zprávy” na stránce 1164
- “Jak ukázka používá synchronizační body” na stránce 1164
- “Jak ukázka používá informace o kontextu zprávy” na stránce 1165
- “Použití identifikátorů zprávy a korelace v CAM” na stránce 1165

Samostatné fronty dotazů a odpovědí v CAM

Aplikace mohla používat jednu frontu pro dotazy i odpovědi, ale byla navržena tak, aby používala samostatné fronty z následujících důvodů:

- Když program zpracovává maximální počet dotazů, další dotazy mohou být ponechány ve frontě. Je-li použita jedna fronta, je třeba ji z fronty vyřadit a uložit ji jinam.
- Ostatní instance CAM by mohly být spuštěny automaticky, aby obsluhují stejnou frontu dotazů, pokud by byl provoz zpráv dostatečně vysoký na to, aby to zaručil. Program však musí sledovat probíhající dotazy, a aby tak mohl učinit, musí dostat zpět všechny odpovědi na dotazy, které zahájil. Pokud se použije pouze jedna fronta, program bude muset procházet zprávy, aby zjistil, zda jsou pro tento program nebo pro jiný. Díky tomu by byla operace mnohem méně efektivní.

  Aplikace může podporovat více modulů CAM a efektivně obnovit probíhající dotazy pomocí párových front pro odpovědi a čekající fronty.

- Program může efektivně čekat ve více frontách pomocí signalizace.

Jak ukázka zpracovává chyby

Program uživatelského rozhraní zpracovává chyby tak, že je přímo nahlásí uživateli.

Ostatní programy nemají uživatelská rozhraní, takže musí zacházet s chybami jinými způsoby. V mnoha situacích (například v případě selhání volání MQGET) tyto jiné programy neznají identitu uživatele aplikace.

Ostatní programy vkládají chybové zprávy do fronty dočasného úložiště CICS s názvem CSQ4SAMP. Tuto frontu můžete procházet pomocí transakce CEBR dodané produktem CICS. Programy také zapisují chybové zprávy do protokolu CICS CSML.

Jak ukázka zpracovává neočekávané zprávy

Při návrhu aplikace fronty zpráv se musíte rozhodnout, jak zpracovat zprávy, které dorazí do fronty neočekávaně.

Dvě základní volby jsou:

- Aplikace již nepracuje, dokud nezpracuje neočekávanou zprávu. To pravděpodobně znamená, že aplikace oznámí operátorovi, ukončí se a zajistí, že se nerestartuje automaticky (může to udělat nastavením spouštění vypnuto). Tato volba znamená, že veškeré zpracování pro aplikaci může být zastaveno jedinou neočekávanou zprávou a k restartování aplikace je nutný zásah operátora.
- Aplikace odebere 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.

Pokud zvolíte druhou možnost:

- Operátor nebo jiný program by měl zkontrolovat zprávy, které jsou vloženy do fronty nedoručených zpráv, aby zjistil, odkud zprávy přicházejí.
- Pokud zprávu nelze vložit 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 zpracuje všechny dotazy s minimálním dopadem z vnějších aktivit, používá ukázková aplikace Credit Check druhou možnost. Chcete-li zachovat ukázku odděleně od jiných aplikací, které používají stejného správce front, ukázka Credit Check nepoužívá frontu nedoručených zpráv systému; místo toho používá vlastní frontu nedoručených zpráv. Tato fronta má název CSQ4SAMP.DEAD.QUEUE. Ukázka ořízne všechny zprávy, které jsou delší než oblast vyrovnávací paměti poskytovaná pro ukázkové programy. K procházení zpráv v této frontě můžete použít ukázkovou aplikaci Procházet, nebo můžete použít ukázkovou aplikaci Tisk zpráv k vytištění zpráv spolu s jejich deskriptory zpráv.

Pokud však rozšiřujete ukázku tak, aby byla spuštěna v rámci více než jednoho správce front, může správce front umístit neočekávané zprávy nebo zprávy, které nelze doručit, do fronty nedoručených zpráv systému.

Jak ukázka používá synchronizační body

Programy v ukázkové aplikaci Credit Check deklarují synchronizační body, aby zajistily, že:

- V odpovědi na každou očekávanou zprávu je odeslána pouze jedna zpráva odpovědi.
- Do fronty nedoručených zpráv vzorku se nikdy nevloží více kopií neočekávaných zpráv.
- CAM může obnovit stav všech částečně dokončených dotazů získáním trvalých zpráv ze své čekající fronty

K dosažení tohoto cíle se používá jedna jednotka práce, která pokrývá získání zprávy, zpracování této zprávy a všechny následné operace vložení.

Jak ukázka používá informace o kontextu zprávy

Když program uživatelského rozhraní (CSQ4CVB1) odesílá zprávy, používá volbu MQPMO_DEFAULT_CONTEXT. To znamená, že správce front generuje informace o identitě i o původním kontextu. Správce front získá tyto informace z transakce, která spustila program (MVB1), a z ID uživatele, který spustil transakci.

Když CAM odesílá dotazové zprávy, používá volbu MQPMO_PASS_IDENTITY_CONTEXT. To znamená, že informace o kontextu identity vkládané zprávy jsou zkopírovány z kontextu identity původní dotazové zprávy. Při použití této volby jsou informace o původním kontextu generovány správcem front.

Když CAM odesílá zprávy s odpovědí, používá volbu MQPMO_ALTERNATE_USER_AUTHORITY. To způsobí, že správce front použije alternativní ID uživatele pro kontrolu zabezpečení, když CAM otevře frontu pro odpověď. CAM používá ID uživatele zadavatele původní dotazové zprávy. To znamená, že uživatelé mohou vidět odpovědi pouze na ty dotazy, které zaslali. Alternativní ID uživatele je získáno z informací o kontextu identity v deskriptoru zprávy původní dotazové zprávy.

Když programy dotazů (CSQ4CVB3/4/5) odesílají zprávy odpovědí, používají volbu MQPMO_PASS_IDENTITY_CONTEXT. To znamená, že informace o kontextu identity vkládané zprávy jsou zkopírovány z kontextu identity původní dotazové zprávy. Při použití této volby jsou informace o původním kontextu generovány správcem front.

Poznámka: ID uživatele přidružené k transakcím MVB3/4/5 vyžaduje přístup k B2.REPLY.n front. Tato ID uživatelů nemusí být stejná jako ta, která jsou přidružena ke zpracovávanému požadavku. Chcete-li obejít toto možné ohrožení zabezpečení, mohou dotazovací programy 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 zprávy a korelace v CAM

Aplikace musí kdykoli sledovat průběh všech aktivních dotazů, které zpracovává. Za tímto účelem používá jedinečný identifikátor zprávy každé zprávy žádosti o úvěr k přidružení všech informací, které má o každém dotazu.

CAM zkopíruje `MsgId` dotazové zprávy do `CorrelId` všech zpráv požadavků, které odešle pro tento dotaz. Ostatní programy v ukázce (CSQ4CVB3 -5) kopírují `CorrelId` každé zprávy, kterou obdrží, do `CorrelId` své zprávy odpovědi.

z/OS *Ukázka kontroly kreditu s více správci front v systému z/OS*

Ukázkovou aplikaci Credit Check můžete použít k demonstraci distribuovaného řazení do front instalací ukázky do dvou správců front a systémů CICS (s každým správcem front připojeným k jinému systému CICS).

Po instalaci ukázkového programu a spuštění monitoru spouštěčů (CKTI) v jednotlivých systémech je třeba provést následující akce:

1. Nastavte komunikační propojení mezi dvěma správci front. Informace, jak to provést, naleznete v tématu Konfigurace distribuovaného řazení do front.
2. V jednom správci front vytvořte lokální definici pro každou vzdálenou frontu (v druhém správci front), kterou chcete použít. Tyto fronty mohou být libovolnou z front CSQ4SAMP.B *n*.ZPRÁVY, kde *n* je 3, 5, 6 nebo 7. (Jedná se o fronty, které jsou obsluhovány programem checking-account a programem agency-query.) Informace o tom, jak to provést, viz DEFINE QREMOTE a Fronty DEFINE.
3. Změňte definici seznamu názvů (CSQ4SAMP.B4.NAMELIST) tak, aby obsahoval názvy vzdálených front, které chcete použít. Informace o tom, jak to provést, viz DEFINE NAMELIST.

z/OS *Rozšíření IMS k ukázce kontroly kreditu na webu z/OS*

Verze programu kontrolního účtu je dodávána jako program zpracování dávkových zpráv (BMP) systému IMS . Je psáno v jazyce C.

Program provádí stejnou funkci jako verze CICS , kromě toho, že pro získání informací o účtu program čte databázi IMS místo souboru VSAM. Pokud nahradíte verzi CICS programu checking-account verzí IMS , neuvidíte žádný rozdíl v metodě použití aplikace.

Chcete-li připravit a spustit verzi IMS , musíte:

1. Postupujte podle pokynů v části “Příprava a spuštění ukázky kontroly kreditu na webu z/OS” na stránce 1156.
2. Postupujte podle pokynů v části “Příprava ukázkové aplikace pro prostředí IMS na systému z/OS” na stránce 1138.
3. Změňte definici alias fronty CSQ4SAMP.B2.OUTPUT.ALIAS pro vyřešení do fronty CSQ4SAMP.B3.IMS.MESSAGES (místo CSQ4SAMP.B3.MESSAGES). Chcete-li to provést, můžete použít jednu z následujících možností:
   - Operační a ovládací panely IBM MQ for z/OS
   - Příkaz ALTER QALIAS .

Dalším způsobem použití programu pro kontrolu účtu IMS je zajistit, aby obsluhovala jednu z front, které přijímají zprávy z distribučního programu. V doručené podobě ukázkové aplikace Credit Check existují tři z těchto front (B5/6/7.MESSAGES), všechny obsluhované programem agency-query. Tento program vyhledává datovou sadu VSAM. Chcete-li porovnat použití datové sady VSAM a databáze IMS , můžete místo toho nastavit, aby program kontrolních účtů IMS obsluhovala jednu z těchto front. Chcete-li to provést, musíte změnit definici seznamu názvů CSQ4SAMP.B4.NAMELIST nahradí jeden z CSQ4SAMP.B *n*.Fronty MESSAGES s CSQ4SAMP.B3.IMS.Fronta ZPRÁV. Můžete použít jednu z následujících možností:

- Operační a ovládací panely IBM MQ for z/OS
- Příkaz ALTER NAMELIST .

Poté můžete spustit ukázku z CICS transakce MVB1. Uživatel nevidí žádný rozdíl v operaci nebo odezvě. Produkt IMS BMP se zastaví buď po přijetí zprávy o zastavení, nebo po pěti minutách nečinnosti.

Návrh programu kontrolního účtu IMS (CSQ4ICB3)

Tento program se spouští jako BMP. Spusťte program pomocí jeho JCL před odesláním jakýchkoli zpráv IBM MQ .

Program vyhledá v databázi IMS číslo účtu ve zprávách o žádosti o úvěr. Načte odpovídající název účtu, průměrný zůstatek a index úvěrové způsobilosti.

Program odešle výsledky hledání v databázi do fronty pro odpověď, která je uvedena ve zpracovávané zprávě IBM MQ . Vrácená zpráva připojí typ účtu a výsledky hledání k přijaté zprávě, aby transakce sestavující odezvu mohla potvrdit, že se zpracovává správný dotaz. Zpráva je ve formě tří 79-znakových skupin, takto:

```
'Response from CHECKING ACCOUNT for name : JONES J B'
'   Opened 870530, 3-month average balance = 000012.57'
'   Credit worthiness index - BBB'
```

Je-li spuštěn jako BMP orientovaný na zprávy, program vyprázdní frontu zpráv IMS , pak přečte zprávy z fronty IBM MQ for z/OS a zpracuje je. Z fronty zpráv IMS nejsou přijaty žádné informace. Program se znovu připojí ke správci front po každém kontrolním bodu, protože manipulátory byly zavřeny.

Při spuštění v dávkově orientovaném BMP je program i nadále připojen ke správci front po každém kontrolním bodu, protože popisovače nejsou zavřené.

z/OS *Ukázka obslužné rutiny zpráv v adresáři z/OS*

Ukázková aplikace TSO obslužné rutiny 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 vzorek podle popisu v části “Příprava ukázkových aplikací pro prostředí TSO na systému z/OS” na stránce 1132.
2. Upravte parametr CLIST (CSQ4RCH1) poskytnutý pro ukázku tak, aby definoval umístění panelů, umístění souboru zpráv a umístění zaváděcích modulů.

Můžete použít CLIST CSQ4RCH1 ke spuštění jak verze C, tak verze COBOL ukázky. Dodávaná verze produktu CSQ4RCH1 spouští verzi jazyka C a obsahuje pokyny pro přizpůsobení potřebné pro verzi jazyka COBOL.

Poznámka:

1. S ukázkou nejsou poskytnuty žádné definice ukázkové fronty.
2. VS COBOL II nepodporuje multitasking s ISPF, proto nepoužívejte ukázkovou aplikaci obslužné rutiny zpráv na obou stranách rozdělené obrazovky. Pokud tak učiníte, výsledky jsou nepředvídatelné.

z/OS *Použití ukázky obslužné rutiny zpráv v systému z/OS*

Po instalaci ukázky a jejím vyvolání z přizpůsobeného rozhraní CLIST CSQ4RCH1se zobrazí obrazovka zobrazená v souboru Obrázek 146 na stránce 1167 .

```
----------------------- 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 146. Počáteční obrazovka pro ukázku obslužné rutiny zpráv

Zadejte správce front a název fronty, který má být zobrazen (s rozlišením velkých a malých písmen), a zobrazí se obrazovka se seznamem zpráv (viz Obrázek 147 na stránce 1168).

```
---------------------- 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 147. Obrazovka se seznamem zpráv pro ukázku obslužné rutiny zpráv

Na této obrazovce je zobrazeno prvních 99 zpráv ve frontě a pro každou z nich jsou zobrazena 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 vložení zprávy do fronty (GMT)

Název formátu
MQMD.Format

Identifikátor uživatele
MQMD.UserIdentifier

Typ vkládající aplikace
MQMD.PutApplType typu PutApplType

Název vkládající aplikace
MQMD.PutApplName

Zobrazí se také celkový počet zpráv ve frontě.

Z této obrazovky lze vybrat zprávu, podle čísla, ne podle pozice kurzoru, a pak zobrazit. Příklad viz Obrázek 148 na stránce 1169.

```
---------------------- 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 148. Zobrazí se zvolená zpráva

Jakmile je zpráva zobrazena, může být odstraněna, ponechána ve frontě nebo postoupena do jiné fronty. Pole `Forward to Q Mgr` a `Forward to Queue` jsou inicializována s hodnotami z deskriptoru MQMD. Před postoupením zprávy je možné je změnit.

Vzorový návrh umožňuje výběr a zobrazení pouze zpráv s jedinečnými kombinacemi `MsgId` / `CorrelId` , protože zpráva je načtena pomocí `MsgId` a `CorrelId` jako klíče. Není-li klíč jedinečný, ukázka nemůže s jistotou načíst vybranou zprávu.

Poznámka: Použijete-li k procházení zpráv ukázku SCSQCLST (CSQ4RCH1), každé vyvolání způsobí zvýšení počtu vrácení zprávy. Chcete-li změnit chování této ukázky, zkopírujte ukázku a podle potřeby upravte její obsah. Měli byste si uvědomit, že další aplikace, které spoléhají na tento počet vrácení, mohou být ovlivněny tímto zvyšujícím se počtem.

z/OS *Návrh ukázky obslužné rutiny zpráv v systému z/OS*

Toto téma popisuje návrh jednotlivých programů, které tvoří ukázkovou aplikaci obslužné rutiny zpráv.

Program pro ověření platnosti objektů

To vyžaduje platný název fronty a správce front.

Pokud nezadáte název správce front, použije se výchozí správce front, je-li k dispozici. Lze použít pouze lokální fronty. Je vydán příkaz MQINQ, který kontroluje, zda je typ fronty a chyba ohlášena, pokud fronta není lokální. Není-li fronta úspěšně otevřena nebo je-li ve frontě zablokováno volání MQGET, vrátí se chybové zprávy označující CompCode a návratový kód příčiny.

Program seznamu zpráv

Zobrazí se seznam zpráv ve frontě s informacemi o nich, jako např. datum putdate, čas puttime a formát zprávy.

Maximální počet zpráv uložených v seznamu je 99. Pokud je ve frontě více zpráv, zobrazí se také aktuální hloubka fronty. Chcete-li vybrat zprávu pro zobrazení, zapište číslo zprávy do vstupního pole (předvolba je 01). Pokud váš záznam není platný, obdržíte odpovídající chybovou zprávu.

Program pro obsah zpráv

Zobrazí se obsah zprávy.

Obsah je formátován a rozdělen do dvou částí:

1. deskriptor zprávy
2. Vyrovnávací paměť zprávy

Deskriptor zprávy zobrazuje obsah každého pole na samostatném řádku.

Vyrovnávací paměť zpráv je formátována v závislosti na jejím obsahu. Pokud vyrovnávací paměť obsahuje záhlaví nedoručených zpráv (MQDLH) nebo záhlaví přenosové fronty (MQXQH), jsou tyto zprávy formátovány a zobrazeny před samotnou vyrovnávací pamětí.

Před formátováním dat vyrovnávací paměti zobrazuje řádek titulku délku vyrovnávací paměti zprávy v bajtech. Maximální velikost vyrovnávací paměti je 32768 bajtů a jakákoli zpráva delší, než je tato velikost, je oříznuta. Zobrazí se plná velikost vyrovnávací paměti spolu se zprávou, která označuje, že se zobrazí pouze prvních 32768 bajtů zprávy.

Data vyrovnávací paměti jsou formátována dvěma způsoby:

1. Po vytištění offsetu do vyrovnávací paměti jsou data vyrovnávací paměti zobrazena hexadecimálně.
2. Data vyrovnávací paměti se pak znovu zobrazí jako hodnoty EBCDIC. Pokud nelze vytisknout žádnou hodnotu EBCDIC, vytiskne místo toho tečku (.).

Můžete zadat D pro odstranění, nebo F pro postoupení do pole akce. Pokud zvolíte postoupení zprávy, `forward-to queue` a `queue manager name` musí být správně nastaveny. Výchozí nastavení pro tato pole jsou načtena z polí deskriptoru zpráv ReplyToQ a ReplyToQMgr.

Pokud předáte zprávu, jakýkoli blok záhlaví uložený ve vyrovnávací paměti se odkládá. Pokud je zpráva úspěšně postoupena, je odebrána z původní fronty. Pokud zadáte neplatné akce, zobrazí se chybové zprávy.

K dispozici je také příklad panelu nápovědy s názvem CSQ4CHP9 .

z/OS *Ukázka asynchronního vložení na z/OS*

Ukázkový program asynchronního vložení vkládá zprávy do fronty pomocí asynchronního volání MQPUT. Ukázka také načte informace o stavu pomocí volání MQSTAT.

Aplikace asynchronního vložení používají tato volání MQI:

- MQCONN
- MQOPEN
- MQPUT
- MQSTAT
- MQCLOSE

- MQDISC

Ukázkové programy jsou dodávány v programovacím jazyce C.

Aplikace asynchronního vložení jsou spuštěny v dávkovém prostředí. Viz Další ukázky pro dávkové aplikace.

Toto téma také poskytuje informace o návrhu programu asynchronní spotřeby a spuštění ukázky CSQ4BCS2 .

- “Spuštění ukázky CSQ4BCS2” na stránce 1171
- “Návrh ukázkového programu Asynchronous Put” na stránce 1171

Spuštění ukázky CSQ4BCS2

Tento ukázkový program zabírá 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 určen správce front, připojí se modul CSQ4BCS2 k výchozímu správci front. Obsah zprávy je poskytován prostřednictvím standardního vstupu (**SYSIN DD**).

Ke spuštění programu existuje ukázkový soubor JCL, který je umístěn v CSQ4BCSP.

Návrh ukázkového programu Asynchronous Put

Program používá volání MQOPEN se zadanými výstupními volbami nebo s volbami MQOO_OUTPUT a MQOO_FAIL_IF_QUIESCING k otevření cílové fronty pro vkládání zpráv.

Pokud program nemůže otevřít frontu, zobrazí chybovou zprávu obsahující kód příčiny vrácený voláním MQOPEN. Chcete-li zachovat jednoduchost programu při tomto a následných voláních MQI, jsou pro mnoho voleb použity výchozí hodnoty.

Pro každý řádek vstupu program načte text do vyrovnávací paměti a použije volání MQPUT s parametrem MQPMO_ASYNC_RESPONSE k vytvoření zprávy datagramu obsahující text tohoto řádku a asynchronně vloží zprávu do cílové fronty. Program pokračuje, dokud nedosáhne konce vstupu, nebo dokud neselže volání MQPUT. Pokud program dosáhne konce vstupu, uzavře frontu pomocí volání MQCLOSE.

Program poté vydá volání MQSTAT, které vrací strukturu MQSTS, a zobrazí zprávy obsahující počet úspěšně vložených zpráv, počet vložených zpráv s varováním a počet selhání.

Poznámka: Chcete-li zjistit, co se stane při zjištění chyby MQPUT voláním MQSTAT, nastavte parametr MAXDEPTH v cílové frontě na nízkou hodnotu.

z/OS *Ukázka Dávková asynchronní spotřeba na systému z/OS*

Ukázkový program CSQ4BCS1 je dodáván v jazyce C a demonstruje použití MQCB a MQCTL k asynchronnímu přijímání zpráv z více front.

Ukázky asynchronní spotřeby se spouštějí v dávkovém prostředí. Viz Další ukázky pro dávkové aplikace.

K dispozici je také ukázka jazyka COBOL, která se spouští v prostředí CICS , viz “Ukázka CICS Asynchronní spotřeba a publikování/odběr na z/OS” na stránce 1173.

Aplikace používají tato volání MQI:

- MQCONN
- MQOPEN

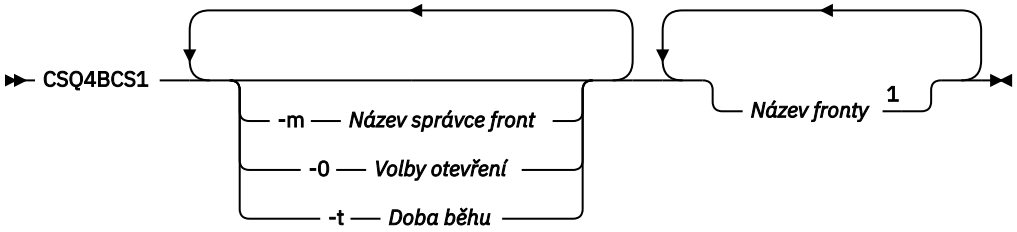
- MQCLOSE
- MQDISC
- MQCB
- MQCTL

V tomto tématu jsou také uvedeny informace o následujících záhlavích:

- “Spuštění ukázky CSQ4BCS1” na stránce 1172
- “Návrh ukázkového programu Dávková asynchronní spotřeba” na stránce 1172

Spuštění ukázky CSQ4BCS1

Tento ukázkový program se řídí následující 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 existuje ukázkový soubor JCL, který je umístěn v adresáři CSQ4BCSC.

Návrh ukázkového programu Dávková asynchronní spotřeba

Ukázka ukazuje, jak číst zprávy z více front v pořadí jejich příchodu. To by vyžadovalo více kódu pomocí synchronního příkazu MQGET. S asynchronní spotřebou se nepožaduje žádný systém výzev a správa podprocesů a úložišť je prováděna produktem IBM MQ. V ukázkovém programu se chyby zapisují do konzoly.

Ukázkový kód má následující kroky:

1. Definujte funkci zpětného volání pro jednotlivou spotřebu 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` nemusí být nastaveno pro každou frontu; jedná se o pole pouze pro vstup. 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 na z/OS*

Ukázkové programy Asynchronní spotřeba a Publikování/odběr demonstrují použití asynchronní spotřeby a funkcí publikování a odběru v rámci produktu CICS.

Program *Registrační klient* registruje tři obslužné rutiny zpětného volání (obslužnou rutinu událostí a dva spotřebitele zpráv) a spustí asynchronní spotřebu. Program *Klient systému zpráv* vkládá zprávy do fronty nebo publikuje vhodné zprávy z konzoly CICS pro spotřebu dvěma spotřebiteli zpráv (CSQ4CVCN a CSQ4CVCT).

Chcete-li poskytnout běhovou kontrolu nad chováním ukázky, může být jeden ze spotřebitelů zpráv instruován pomocí zpráv, které obdrží, k SUSPEND, RESUME nebo DEREGISTER s některou z obslužných rutin zpětného volání. Lze jej také použít k vydání příkazu MQCTL STOP pro ukončení řízené asynchronní spotřeby. Další spotřebitel zpráv je registrován k odběru tématu.

Každý program zadá příkazy COBOL DISPLAY v příslušných bodech, aby zobrazil chování ukázky.

Aplikace používají tato volání MQI:

- MQOPEN
- MQPUT
- MQSUB
- MQGET
- MQCLOSE
- MQCB
- MQCTL

Programy jsou dodávány v jazyce COBOL. Viz CICS Ukázky asynchronní spotřeby a publikování/odběru pro aplikace CICS .

Toto téma také poskytuje následující informace:

- “Nastavení” na stránce 1173
- “Klient registrace CSQ4CVRG” na stránce 1174
- “Obslužná rutina událostí CSQ4CVEV” na stránce 1174
- “Spotřebitel jednoduchých zpráv CSQ4CVCN” na stránce 1174
- “Spotřebitel řídicích zpráv CSQ4CVCT” na stránce 1174
- “Klient systému zpráv CSQ4CVPT” na stránce 1174

Nastavení

Názvy fronty a tématu používané spotřebiteli zpráv jsou pevně naprogramovány v programech klienta pro registraci a zasílání zpráv.

Fronta, **SAMPLE.CONTROL.QUEUE**, měla by být definována pro správce front přidruženého k oblasti CICS před spuštěním ukázky. Téma **Novinky/Média/filmy**lze v případě potřeby definovat, nebo je vytvořeno za běhu pod výchozím administrativním objektem, pokud neexistuje.

Programy CICS a definice transakcí lze nainstalovat instalací skupiny: CSQ4SAMP.

Klient registrace CSQ4CVRG

Program klienta registrace musí být spuštěn v rámci transakce CICS MVRG. Nepotřebuje žádný vstup.

Při spuštění Registrační klient registruje následující obslužné rutiny zpětného volání pomocí MQCB:

- CSQ4CVEV jako obslužná rutina událostí.
- CSQ4CVCN jako spotřebitel zpráv na téma **Novinky/Média/filmy**.
- CSQ4CVCT jako spotřebitel zpráv ve frontě, **SAMPLE.CONTROL.QUEUE**.

Klient registrace předá datovou strukturu obsahující názvy všech tří registrovaných obslužných rutin zpětného volání do CSQ4CVCTspolu s manipulátory objektů přidruženými k těmto dvěma odběratelům zpráv.

Po registraci obslužných rutin zpětného volání klient registrace vydá příkaz MQCTL START_WAIT ke spuštění asynchronní spotřeby a pozastaví se, dokud se mu nevrátí řízení (například jedním z obslužných rutin zpětného volání vydávajících příkaz MQCTL STOP).

Obslužná rutina událostí CSQ4CVEV

Je-li řízena, obslužná rutina událostí zobrazí zprávu označující typ volání (například START). Je-li obslužná rutina událostí řízena pro kód příčiny IBM MQ CONNECTION_QUIESCING, vydá příkaz MQCTL STOP pro ukončení asynchronní spotřeby a vrátí řízení klientovi registrace.

Spotřebitel jednoduchých zpráv CSQ4CVCN

Je-li tento spotřebitel zpráv řízen, zobrazí zprávu označující typ volání (například REGISTER). Je-li řízen pro typ volání MSG_REMOVED, spotřebitel zpráv načte příchozí zprávu a odešle ji do protokolu úlohy CICS .

Spotřebitel řídicích zpráv CSQ4CVCT

Je-li tento spotřebitel zpráv řízen, zobrazí zprávu označující typ volání (například START). Je-li řízen pro typ volání MSG_REMOVED, spotřebitel zpráv načte příchozí zprávu a datovou strukturu předanou klientem registrace. Na základě obsahu zprávy vydá příslušné příkazy MQCB nebo MQCTL pro jednu z následujících možností:

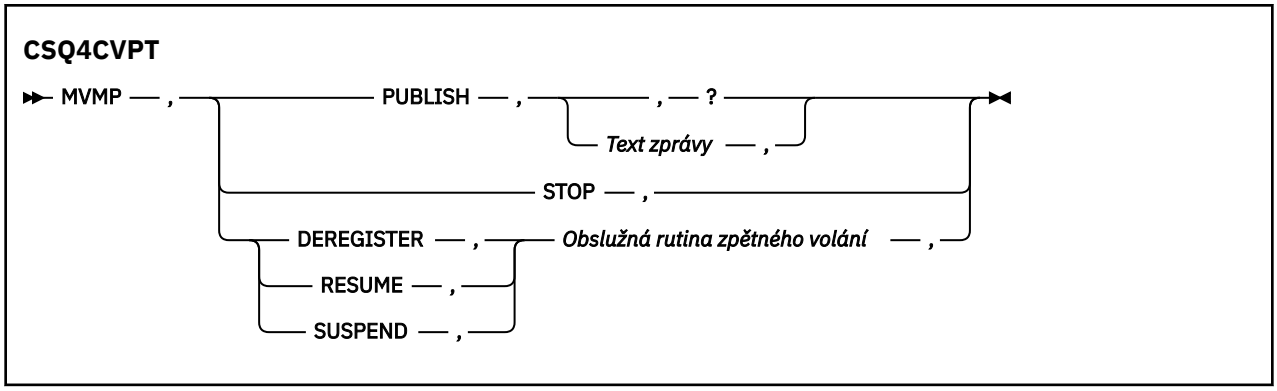
- STOP Asynchronní spotřeba (vrácení řízení registračnímu klientovi).
- SUSPEND, RESUME nebo DEREGISTER pojmenovanou obslužnou rutinu zpětného volání (včetně sebe).

Klient systému zpráv CSQ4CVPT

Klient systému zpráv má dvě funkce:

- Publikuje zprávu do tématu pro spotřebu konzumentem zpráv CSQ4CVCN.
- Vkládá řídicí zprávu do fronty pro spotřebu konzumentem řídicích zpráv CSQ4CVCT, což má za následek potenciální změnu chování ukázky.

Program klienta systému zpráv musí být spuštěn z konzoly CICS v rámci transakce CICS a používá vstup příkazového řádku s následující syntaxí:



PUBLISH
Publikujte text zprávy (nebo výchozí zprávu) jako zadrženou zprávu pro spotřebu konzumentem jednoduchých zpráv.

ZASTAVIT
Zastavit asynchronní spotřebu.

DEREGISTER (REGISTROVAT)
Zrušit registraci jmenované obslužné rutiny zpětného volání.

RESUME
Obnovte uvedenou obslužnou rutinu zpětného volání.

SUSPEND
Pozastavte uvedenou obslužnou rutinu zpětného volání.

Vstupní pole jsou poziční a oddělená čárkami. Klíčová slova a názvy obslužné rutiny zpětného volání nerozlišují velká a malá písmena.

Příklady:

Tabulka 186. Příklady vstupu

Příklad	**Popis**
MVMP, PUBLIKOVAT,,	Publikovat výchozí zprávu
MVMP,publikovat, `A short message`,	Publikovat daný text
MVMP, STOP,	Zastavit asynchronní spotřebu
MVMP,DEREGISTER,CSQ4CVEV,	Zrušit registraci obslužné rutiny událostí
MVMP,resume,csq4cvcn,	Pokračovat v konzumentu 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 zpětného volání ukončí START_WAIT vydaný klientem registrace, vrátí mu řízení a ukončí úlohu.
- Pozastavení nebo zrušení registrace obslužné rutiny zpětného volání řízení nebylo záměrně zabráněno, ale odebere schopnost dále řídit chování ukázky.

z/OS *Ukázka publikování/odběru na webu z/OS*

Ukázkové programy publikování/odběru demonstrují použití funkcí publikování a odběru v produktu IBM MQ.

Existují čtyři ukázkové programy programovacího jazyka C a dva programy programovacího jazyka COBOL, které demonstrují, jak programovat do rozhraní publikování/odběru IBM MQ . Programy jsou dodávány

v jazyce C a COBOL. Aplikace se spouštějí v dávkovém prostředí; viz Ukázky publikování/odběru pro dávkové aplikace.

Existují také ukázky jazyka COBOL, které se spouštějí v prostředí CICS ; viz “Ukázka CICS Asynchronní spotřeba a publikování/odběr na z/OS” na stránce 1173.

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 jazyce C; publikuje zprávy do tématu. Před spuštěním tohoto programu spusťte jednu 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 určen správce front, připojí se modul CSQ4BCP1 k výchozímu správci front. Pro spuštění programu je k dispozici ukázkový soubor JCL, který je umístěn v adresáři CSQ4BCPP.

Obsah zprávy je poskytován prostřednictvím standardního vstupu (**SYSIN DD**).

Spuštění ukázky CSQ4BCP2

Tento program je napsán v jazyce C; přihlásí se k odběru tématu a vytiskne přijaté zprávy.

Tento program má až tři parametry:

1. Název cílového řetězce tématu (povinné).
2. Název správce front (volitelné).
3. Volby odběru MQSD (volitelné).

Není-li určen správce front, připojí se modul CSQ4BCP2 k výchozímu správci front. Pro spuštění programu existuje ukázkový soubor JCL, který je umístěn v adresáři CSQ4BCPS.

Spuštění ukázky CSQ4BCP3

Tento program je napsán v jazyce C; přihlásí se k odběru tématu pomocí uživatelem určené cílové fronty a vytiskne 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 určen správce front, připojí se modul CSQ4BCP3 k výchozímu správci front. Pro spuštění programu je k dispozici ukázkový soubor JCL, který je umístěn v adresáři CSQ4BCPD.

Spuštění ukázky CSQ4BCP4

Tento program je napsán v jazyce C; odebírá a získává zprávy z tématu, které umožňuje použití rozšířených voleb pro volání MQSUB a rozšiřuje ty, které jsou k dispozici v jednodušší ukázce MQSUB: CSQ4BCP2. Kromě informačního obsahu zprávy jsou přijaty a zobrazeny vlastnosti zprávy pro každou zprávu.

Tento program přebírá sadu proměnných parametrů:

- **-t** *Topic string*.
- **-o** *Topic object name*.

  Důležité: Je požadován jeden z **-t** nebo **-o**, nebo obojí.
- **-m** *Queue manager name* (volitelné).
- **-b** *Connection binding type* (volitelné), kde *typ* může mít libovolnou z následujících hodnot:
  - *standard* : MQCNO_STANDARD_BINDING, což je výchozí hodnota.
  - *shared*: MQCNO_SHARED_BINDING.
  - *fastpath*: MQCNO_FASTPATH_BINDING.
  - *isolated*: MQCNO_ISOLATED_BINDING
- **-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
  - Není zadána žádná hodnota: Není-li zadána žádná hodnota, výchozí hodnota je 30 sekund.
- **-d** *Subscription name* (volitelné). Vytvoří nebo obnoví pojmenovaný trvalý odběr.
- **-k** (volitelné). Uchovává trvalý odběr na MQCLOSE.

Není-li určen správce front, připojí se modul CSQ4BCP4 k výchozímu správci front. Pro spuštění programu je k dispozici ukázkový soubor JCL, který je umístěn v adresáři CSQ4BCPE.

Spuštění ukázky CSQ4BVP1

Tento program je napsán v jazyce COBOL, publikuje zprávy do tématu. Před spuštěním tohoto programu spusťte jednu z ukázek odběratele.

Tento program nepřijímá žádné parametry. Produkt **SYSIN DD** poskytuje název vstupního tématu, název správce front a obsah zprávy.

Není-li určen správce front, připojí se modul CSQ4BVP1 k výchozímu správci front. Pro spuštění programu je k dispozici ukázkový soubor JCL, který je umístěn v adresáři CSQ4BVPP.

Spuštění ukázky CSQ4BVP2

Tento program je napsán v jazyce COBOL, přihlásí se k odběru tématu a vytiskne přijaté zprávy.

Tento program nepřijímá žádné parametry. Produkt **SYSIN DD** poskytuje vstup pro název tématu a název správce front.

Není-li určen správce front, připojí se modul CSQ4BVP1 k výchozímu správci front. Pro spuštění programu je k dispozici ukázkový soubor JCL, který je umístěn v adresáři CSQ4BVPP.

z/OS *Ukázka vlastnosti zprávy Set and Inquire v systému z/OS*

Ukázkové programy vlastností zprávy demonstrují přidání vlastností definovaných uživatelem do popisovače zprávy a inkvizici vlastností přidružených k této zprávě.

Aplikace používají tato volání MQI:

- MQCONN
- MQOPEN
- MQPUT
- MQGET
- MQCLOSE
- MQDISC
- MQCRTMH
- MQDLTMH
- MQINQMP
- MQSETMP

Programy jsou dodávány v jazyce C. Aplikace se spouštějí v dávkovém prostředí. Viz Další ukázky pro dávkové aplikace.

Program CSQ4BCM1 se používá k dotazování vlastností manipulátoru zprávy z fronty zpráv a je příkladem použití volání rozhraní API MQINQMP. Ukázka získá jednu zprávu z fronty a pak vytiskne všechny vlastnosti popisovače zprávy.

Program CSQ4BCM2 se používá k nastavení vlastností popisovače zprávy 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 vlastností dotazování a tisku zpráv jsou obsaženy v ukázkových programech CSQ4BCG1 a CSQ4BCP4 .

Toto téma také poskytuje informace o spuštění ukázek vlastností Nastavit a dotazovat se pod následujícími nadpisy:

- “Spuštění ukázky CSQ4BCM1” na stránce 1178
- “Spuštění ukázky CSQ4BCM2” na stránce 1178

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 zpráv jsou poskytovány prostřednictvím standardního vstupu (**SYSIN DD**). Ke spuštění programu existuje ukázkový soubor JCL, který je umístěn v adresáři CSQ4BCMP.

Vyvíjení aplikací pro Managed File Transfer

Uveďte programy, které se mají spustit s produktem Managed File Transfer, použijte Apache Ant s Managed File Transfer, přizpůsobte Managed File Transfer s uživatelskými uživatelskými programy a řídte Managed File Transfer vložením zpráv do fronty příkazů agenta.

Určení programů, které se mají spustit s MFT

Programy můžete spustit na systému, kde je spuštěn Managed File Transfer Agent . Jako součást požadavku na přenos souborů můžete uvést program, který se má spustit buď před zahájením přenosu, nebo po jeho dokončení. Dále můžete spustit program, který není součástí požadavku na přenos souborů, odesláním požadavku na spravované volání.

Informace o této úloze

Existuje pět scénářů, ve kterých můžete určit program, který se má spustit:

- Jako součást požadavku na přenos, na zdrojovém agentovi, před spuštěním přenosu.
- Jako součást požadavku na přenos, na cílovém agentu, před spuštěním přenosu.
- Jako součást požadavku na přenos, na zdrojovém agentovi, po dokončení přenosu.
- Jako součást požadavku na přenos na cílovém agentovi po dokončení přenosu.
- Není součástí požadavku na přenos. Můžete odeslat požadavek agentovi ke spuštění programu. Tento scénář je někdy označován jako spravované volání.

Uživatelské procedury a volání programu jsou vyvolána v následujícím pořadí:

```
- SourceTransferStartExit(onSourceTransferStart).
- PRE_SOURCE Command.
- DestinationTransferStartExits(onDestinationTransferStart).
- PRE_DESTINATION Command.
- The Transfer request is performed.
- DestinationTransferEndExits(onDestinationTransferENd).
- POST_DESTINATION Command.
- SourceTransferEndExits(onSourceTransferEnd.
- POST_SOURCE Command.
```

Notes:

1. **`DestinationTransferEndExits`** se spustí pouze po dokončení přenosu, buď úspěšně, nebo částečně úspěšně.
2. **`postDestinationCall`** se spustí pouze po dokončení přenosu, buď úspěšně, nebo částečně úspěšně.
3. **`SourceTransferEndExits`** se spustí pro úspěšné, částečně úspěšné nebo nezdařené přenosy.
4. **`postSourceCall`** se volá pouze v případě, že:
   - Přenos nebyl zrušen.
   - Existuje úspěšný nebo částečně úspěšný výsledek.
   - Všechny programy přenosu po cíli byly úspěšně spuštěny.

Procedura

- Uveďte program, který chcete spustit, pomocí jedné z následujících voleb:

  Použít úlohu Apache Ant

  Ke spuštění programu použijte jednu z úloh `fte:filecopy`, `fte:filemovea` `fte:call` Ant . Pomocí úlohy Ant můžete určit program v libovolném z pěti scénářů pomocí vnořených prvků `fte:presrc`, `fte:predst`, `fte:postdst`, `fte:postsrca` `fte:command` . Další informace viz vnořené prvky vyvolání programu.

Upravit zprávu požadavku na přenos souborů

Můžete upravit XML, který je generován požadavkem na přenos. Pomocí této metody můžete spustit program v libovolném z pěti scénářů přidáním prvků **preSourceCall**, **postSourceCall**, **preDestinationCall**, **postDestinationCall**a **managedCall** do souboru XML. Poté použijte tento upravený soubor XML jako definici přenosu pro nový požadavek na přenos souborů, například s parametrem **fteCreateTransfer -td** . Další informace viz Příklady zpráv požadavků volání agenta MFT.

Použijte příkaz `fteCreateTransfer`

Pomocí příkazu **fteCreateTransfer** můžete zadat programy, které se mají spustit. Příkaz můžete použít k určení programů, které se mají spustit v prvních čtyřech scénářích, jako součást požadavku na přenos, ale nemůžete spustit spravované volání. Informace o parametrech, které se mají použít, naleznete v tématu **fteCreateTransfer**: Spuštění nového přenosu souborů. Příklady použití tohoto příkazu naleznete v tématu Příklady použití příkazu fteCreateTransfer ke spuštění programů.

Související odkazy

Vlastnost commandPath MFT

Spravovaná volání

Agenti Managed File Transfer (MFT) se obvykle používají k přenosu souborů nebo zpráv. Ty se nazývají *Spravované přenosy*. Agenty lze také použít ke spouštění příkazů, skriptů nebo JCL bez nutnosti přenosu souborů nebo zpráv. Tato schopnost se nazývá *Spravovaná volání*.

Požadavky na spravovaná volání lze odeslat agentovi několika způsoby:

- Pomocí úlohy fte: call Ant task.
- Konfigurace monitoru prostředků s úlohou XML, která spouští příkaz nebo skript. Další informace viz Konfigurace úloh monitorování pro spouštění příkazů a skriptů .
- Přímé vložení zprávy XML do fronty příkazů agenta. Další podrobnosti o schématu XML spravovaného volání viz Formát zprávy požadavku na přenos souborů .

Pro spravovaná volání musí být adresář obsahující spouštěný příkaz nebo skript uveden ve vlastnosti agenta **commandPath**.

Spravovaná volání nemohou spouštět příkazy nebo skripty umístěné v adresářích, které nejsou uvedeny v **commandPath**agenta. Tím zajistíte, že agent nespustí žádný škodlivý kód.

Důležité: Chcete-li se ujistit, že se jedná o tento případ, standardně, když zadáte **commandPath**:

- Jakékoli existující prostředí sandbox agenta je konfigurováno agentem při jeho spuštění, takže všechny adresáře **commandPath** jsou automaticky přidány do seznamu adresářů, které mají odepřený přístup pro přenos.
- Všechna existující uživatelská pískoviště se aktualizují při spuštění agenta, takže všechny adresáře **commandPath** (a jejich podadresáře) se přidají jako prvky `<exclude>` do prvků `<read>` a `<write>` .
- Pokud není agent nakonfigurován pro použití sandboxů agenta nebo sandboxů uživatele, vytvoří se při spuštění agenta nový sandbox agenta, který má adresáře **commandPath** určené jako odepřené adresáře.

Dále můžete také povolit kontrolu oprávnění na agentovi, abyste se ujistili, že pouze autorizovaní uživatelé mohou odesílat požadavky na spravovaná volání. Další informace naleznete v tématu Omezení oprávnění uživatelů na MFT akcích agenta.

Příkaz, skript nebo skript JCL vyvolaný jako součást spravovaného volání se spouští jako externí proces, který je monitorován agentem. Když se proces ukončí, spravované volání se dokončí a návratový kód z procesu se zpřístupní buď agentovi, nebo skriptu Ant , který vyvolal úlohu **fte:call** Ant .

Pokud bylo spravované volání spuštěno úlohou **fte:call** Ant , může váš skript Ant zkontrolovat hodnotu návratového kódu a určit, zda bylo spravované volání úspěšné či nikoli.

Pro všechny ostatní typy spravovaných volání můžete určit, které hodnoty návratového kódu by měly být použity k označení, že spravované volání bylo úspěšně dokončeno. Agent porovná návratový kód procesu s těmito návratovými kódy po dokončení externího procesu.

Poznámka: Protože spravovaná volání běží jako externí procesy, nelze je po spuštění zrušit.

Spravovaná volání a sloty pro přenos zdroje

Agent obsahuje řadu zdrojových slotů pro přenos, jak je uvedeno ve vlastnosti agenta **maxSourceTransfers**, popsané v části Rozšířené vlastnosti agenta: Limit přenosu.

Kdykoli je spuštěno spravované volání nebo spravovaný přenos, zabírají zdrojový přenosový slot. Slot je uvolněn po dokončení spravovaného volání nebo spravovaného přenosu.

Pokud jsou všechny sloty pro přenos zdroje používány, když agent obdrží buď nové spravované volání, nebo požadavek na spravovaný přenos, agent požadavek zařadí do fronty, dokud nebude slot k dispozici.

Pokud spravované volání spustí spravovaný přenos (například pokud spravované volání spustí skript Ant a skript Ant použije k přenosu souboru úlohu fte: filecopy nebo fte: filemove), jsou vyžadovány dva sloty pro přenos zdroje:

- Jedna pro spravovaný přenos
- Jedna pro spravované volání

V této situaci je důležité si uvědomit, že pokud dokončení spravovaného přenosu trvá buď dlouho, nebo jde do obnovy, pak jsou oba sloty zdrojového přenosu obsazeny, dokud se spravovaný přenos nedokončí, nezruší nebo nevymizí kvůli **transferRecoveryTimeout**. Podrobnosti o **transferRecoveryTimeout**naleznete v tématu Koncepty časového limitu obnovy přenosu . To může potenciálně omezit počet jiných spravovaných přenosů nebo spravovaných volání, která může agent zpracovat.

Z tohoto důvodu byste měli zvážit návrh spravovaného volání, abyste se ujistili, že nezabírá sloty pro přenos zdroje po dlouhou dobu.

Použití REST API se spravovanými voláními

V 9.3.0 V 9.3.0

Příkazová slova HTTP GET a HTTP POST jsou podporována pro povolení spravovaných volání a pracují pouze s verzí 3 REST API.

Jiná příkazová slova, například HTTP DELETE a HTTP UPDATE, nejsou podporována a vrátí kód chyby HTTP 405, pokud se je pokusíte použít.

Upozornění: Po odeslání nelze spravované volání zrušit pomocí REST API.

Použití Apache Ant s MFT

Produkt Managed File Transfer poskytuje úlohy, které můžete použít k integraci funkce přenosu souborů do nástroje Apache Ant .

Příkaz **fteAnt** můžete použít ke spuštění úloh Ant v prostředí Managed File Transfer , které jste již nakonfigurovali. Pomocí úloh Ant přenosu souborů ze skriptů Ant můžete koordinovat složité operace přenosu souborů z interpretovaného skriptovacího jazyka.

Další informace o produktu Apache Antviz Apache Ant webová stránka projektu: https://ant.apache.org/

Související pojmy

“Začínáme s použitím skriptů Ant s produktem MFT” na stránce 1182
Použití skriptů Ant s produktem Managed File Transfer vám umožňuje koordinovat složité operace přenosu souborů z interpretovaného skriptovacího jazyka.

fteAnt: spuštění Ant úloh v MFT

Související odkazy

“Ukázkové úlohy Ant pro MFT” na stránce 1183

Při instalaci produktu Managed File Transferje k dispozici řada ukázkových skriptů Ant . Tyto ukázky jsou umístěny v adresáři *MQ_INSTALLATION_PATH*/mqft/samples/fteant. Každý ukázkový skript obsahuje cíl `init` , upravte vlastnosti nastavené v cíli `init` tak, aby tyto skripty spouštěli s vaší konfigurací.

Začínáme s použitím skriptů Ant s produktem MFT

Použití skriptů Ant s produktem Managed File Transfer vám umožňuje koordinovat složité operace přenosu souborů z interpretovaného skriptovacího jazyka.

Skripty Ant

Skripty Ant (nebo soubory sestavení) jsou dokumenty XML definující jeden nebo více cílů. Tyto cíle obsahují prvky úlohy ke spuštění. Produkt Managed File Transfer poskytuje úlohy, které můžete použít k integraci funkce přenosu souborů do produktu Apache Ant. Chcete-li se dozvědět více o skriptech Ant , prohlédněte si webovou stránku projektu Apache Ant : https://ant.apache.org/

Příklady skriptů Ant , které používají úlohy Managed File Transfer , jsou poskytnuty s instalací produktu v adresáři *MQ_INSTALLATION_PATH*/mqft/samples/fteant

V agentech mostu protokolů jsou skripty Ant spuštěny na systému agenta mostu protokolů. Tyto skripty Ant nemají přímý přístup k souborům na serveru FTP nebo SFTP.

Obor názvů

Obor názvů se používá k odlišení Ant úloh přenosu souborů od jiných Ant úloh, které mohou mít stejný název. Obor názvů definujete ve značce projektu svého skriptu Ant .

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns:fte="antlib:com.ibm.wmqfte.ant.taskdefs" default="do_ping">

  <target name="do_ping">
     <fte:ping cmdqm="qm@localhost@1414@SYSTEM.DEF.SVRCONN" agent="agent1@qm1"
      rcproperty="ping.rc" timeout="15"/>
  </target>

</project>
```

Atribut `xmlns:fte="antlib:com.ibm.wmqfte.ant.taskdefs"` sděluje produktu Ant , aby vyhledal definice úloh s předponou `fte` v knihovně `com.ibm.wmqfte.ant.taskdefs`.

Jako předponu oboru názvů nemusíte používat `fte` ; můžete použít libovolnou hodnotu. Předpona oboru názvů `fte` se používá ve všech příkladech a ukázkových skriptech Ant .

Spuštění skriptů Ant

Chcete-li spustit skripty Ant , které obsahují úlohy přenosu souborů Ant , použijte příkaz **fteAnt** . Příklad:

```
fteAnt -file ant_script_location/ant_script_name
```

Další informace viz **fteAnt**: spuštění Ant úloh v MFT.

Návratové kódy

Úlohy Ant přenosu souborů vracejí stejné návratové kódy jako příkazy Managed File Transfer . Další informace viz Návratové kódy pro produkt MFT.

Související odkazy

fteAnt: spuštění Ant úloh v MFT

“Ukázkové úlohy Ant pro MFT” na stránce 1183
Při instalaci produktu Managed File Transferje k dispozici řada ukázkových skriptů Ant . Tyto ukázky jsou umístěny v adresáři *MQ_INSTALLATION_PATH*/mqft/samples/fteant. Každý ukázkový skript obsahuje cíl `init` , upravte vlastnosti nastavené v cíli `init` tak, aby tyto skripty spouštěli s vaší konfigurací.

Ukázkové úlohy Ant pro MFT

Při instalaci produktu Managed File Transferje k dispozici řada ukázkových skriptů Ant . Tyto ukázky jsou umístěny v adresáři *MQ_INSTALLATION_PATH*/mqft/samples/fteant. Každý ukázkový skript obsahuje cíl `init` , upravte vlastnosti nastavené v cíli `init` tak, aby tyto skripty spouštěli s vaší konfigurací.

e-mail

Ukázka e-mail demonstruje, jak použít úlohy Ant k přenosu souboru a odeslání e-mailu na uvedenou e-mailovou adresu, pokud se přenos nezdaří. Skript kontroluje, zda je zdrojový a cílový agent aktivní a zda je schopen zpracovat přenosy pomocí úlohy Managed File Transfer ping . Pokud jsou aktivní oba agenti, skript použije úlohu Managed File Transfer fte: filecopy k přenosu souboru mezi zdrojovým a cílovým agenty bez odstranění původního souboru. Pokud se přenos nezdaří, skript odešle e-mail obsahující informace o selhání pomocí standardní úlohy Ant email .

Centrální server

Ukázka centrálního serveru se skládá ze dvou skriptů: `hubcopy.xml` a `hubprocess.xml.` Skript `hubcopy.xml` ukazuje, jak můžete použít skriptování Ant k sestavení topologií stylu 'hub and Mluvený'. V této ukázce jsou dva soubory přeneseny z agentů spuštěných na paprskových počítačích na agenta spuštěného na centrálním počítači. Oba soubory se přenášejí ve stejnou dobu a po dokončení přenosů se na centrálním počítači spustí skript `hubprocess.xml` Ant , který soubory zpracuje. Pokud se oba soubory přenášejí správně, skript Ant zřetězí obsah souborů. Pokud se soubory nepřenášejí správně, skript Ant se vyčistí odstraněním dat souboru, která byla přenesena. Aby tento příklad fungoval správně, musíte vložit skript `hubprocess.xml` do cesty k příkazům agenta centrálního serveru. Další informace o nastavení cesty k příkazům agenta viz commandPath MFT property.

librarytransfer (pouze platformaIBM i)

IBM i

IBM i Ukázka librarytransfer demonstruje, jak používat úlohy Ant k přenosu knihovny IBM i na jednom systému IBM i do druhého systému IBM i .

IBM i Ukázka librarytransfer používá podporu nativních souborů typu save file v systému IBM i s předdefinovanými úlohami Ant dostupnými v produktu Managed File Transfer pro přenos nativních objektů knihovny mezi dvěma systémy IBM i . Ukázka používá vnořený prvek < presrc> v úloze Managed File Transfer filecopy k vyvolání spustitelného skriptu `librarysave.sh` , který uloží požadovanou knihovnu na systému zdrojového agenta do dočasného souboru typu save file. Soubor typu save file je přesunut úlohou filecopy ant do cílového systému agenta, kde se používá vnořený prvek < postdst> k vyvolání spustitelného skriptu `libraryrestore.sh` k obnově knihovny uložené v souboru typu save file do cílového systému.

IBM i Před spuštěním této ukázky je třeba provést určitou konfiguraci, jak je popsáno v souboru `librarytransfer.xml` . Musíte mít také pracovní prostředí Managed File Transfer na dvou počítačích se systémem IBM i . Nastavení se musí skládat ze zdrojového agenta spuštěného na prvním počítači se systémem IBM i a cílového agenta spuštěného na druhém počítači se systémem IBM i . Oba agenti musí být schopni vzájemně komunikovat.

IBM i Ukázka librarytransfer se skládá z následujících tří souborů:

- `librarytransfer.xml`

- `librarysave.sh` (spustitelný skript < presrc>)
- `librarysave.sh` (< postdst> spustitelný skript)

Ukázkové soubory jsou umístěny v následujícím adresáři: `/QIBM/ProdData/WMQFTE/V7/samples/fteant/ibmi/librarytransfer`

IBM i Chcete-li spustit tuto ukázku, musí uživatel provést následující kroky:

1. Spusťte relaci Qshell. V příkazovém okně IBM i zadejte: STRQSH
2. Změňte adresář na adresář `bin` takto:

```
cd  /QIBM/ProdData/WMQFTE/V7/bin
```

3. Po dokončení požadované konfigurace spusťte ukázku pomocí následujícího příkazu:

```
fteant -f /QIBM/ProdData/WMQFTE/V7/samples/fteant/ibmi/librarytransfer/librarytransfer.xml
```

physicalfiletransfer (pouze platformaIBM i)

IBM i Ukázka fyzického přenosu souborů demonstruje, jak použít úlohy Ant k přenosu zdrojového fyzického nebo databázového souboru z knihovny na jednom systému IBM i do knihovny na druhém systému IBM i .

IBM i Ukázka physicalfiletransfer používá podporu nativních souborů typu save file v systému IBM i s předdefinovanými úlohami Ant dostupnými v produktu Managed File Transfer k přenosu úplných zdrojových fyzických a databázových souborů mezi dvěma systémy IBM i . Ukázka používá vnořený prvek < presrc> v rámci úlohy Managed File Transfer filecopy k vyvolání spustitelného skriptu `physicalfilesave.sh` k uložení požadovaného zdrojového fyzického nebo databázového souboru z knihovny na systému zdrojového agenta do dočasného souboru typu save file. Soubor typu save file je přesunut úlohou filecopy ant do cílového systému agenta, kde je vnořený prvek < postdst> použit k vyvolání spustitelného skriptu `physicalfilerestore.sh` a poté obnoví objekt souboru uvnitř souboru typu save file do určené knihovny v cílovém systému.

IBM i Před spuštěním této ukázky musíte provést určitou konfiguraci, jak je popsáno v souboru `physicalfiletransfer.xml` . Musíte mít také pracovní prostředí Managed File Transfer na dvou systémech IBM i . Nastavení se musí skládat ze zdrojového agenta spuštěného na prvním systému IBM i a cílového agenta spuštěného na druhém systému IBM i . Oba agenti musí být schopni vzájemně komunikovat.

IBM i Ukázka fyzického přenosu souborů se skládá z následujících tří souborů:

- `physicalfiletransfer.xml`
- `physicalfilesave.sh` (spustitelný skript < presrc>)
- `physicalfilerestore.sh` (< postdst> spustitelný skript)

Ukázkové soubory jsou umístěny v následujícím adresáři: `/QIBM/ProdData/WMQFTE/V7/samples/fteant/ibmi/physicalfiletransfer`

IBM i Chcete-li spustit tuto ukázku, musí uživatel provést následující kroky:

1. Spusťte relaci Qshell. V příkazovém okně IBM i zadejte: STRQSH
2. Změňte adresář na adresář `bin` takto:

```
cd  /QIBM/ProdData/WMQFTE/V7/bin
```

3. Po dokončení požadované konfigurace spusťte ukázku pomocí následujícího příkazu:

```
fteant -f /QIBM/ProdData/WMQFTE/V7/samples/fteant/ibmi/physicalfiletransfer/
physicalfiletransfer.xml
```

časový limit

Ukázka timeout demonstruje, jak použít úlohy Ant k pokusu o přenos souboru a ke zrušení přenosu, pokud trvá déle, než je zadaná hodnota časového limitu. Skript zahájí přenos souboru pomocí úlohy Managed File Transfer fte: filecopy . Výsledek tohoto přenosu je odložen. Skript používá úlohu Managed File Transfer fte: awaitVýsledek Ant k čekání na daný počet sekund na dokončení přenosu. Pokud se přenos v daném čase nedokončí, použije se ke zrušení přenosu souborů úloha Managed File Transfer fte: cancel Ant .

vsamtransfer

z/OS

z/OS Ukázka vsamtransfer demonstruje, jak použít úlohy Ant k přenosu z datové sady VSAM do jiné datové sady VSAM pomocí Managed File Transfer. Produkt Managed File Transfer momentálně nepodporuje přenos datových sad VSAM. Ukázkový skript uvolní datové záznamy VSAM do sekvenční datové sady pomocí `presrc` vnořených prvků vyvolání programu pro volání spustitelného souboru `datasetcopy.sh`. Skript používá úlohu Managed File Transfer fte: filemove k přenosu sekvenční datové sady ze zdrojového agenta do cílového agenta. Skript poté použije `postdst` vnořené prvky vyvolání programu k volání skriptu `loadvsam.jcl` . Tento skript JCL načte přenesené záznamy datových sad do cílové datové sady VSAM. Tato ukázka používá JCL pro volání místa určení k předvedení této volby jazyka. Stejného výsledku lze dosáhnout také pomocí druhého skriptu shellu.

z/OS Tato ukázka nevyžaduje, aby zdrojová a cílová datová sada byla VSAM. Ukázka funguje pro všechny datové sady, pokud jsou zdrojové a cílové datové sady stejného typu.

z/OS Aby tato ukázka fungovala správně, musíte vložit skript `datasetcopy.sh` na cestu k příkazům zdrojového agenta a skript `loadvsam.jcl` na cestu k příkazům cílového agenta. Další informace o nastavení cesty k příkazům agenta viz commandPath MFT property.

zip

Ukázka zip se skládá ze dvou skriptů: `zip.xml` a `zipfiles.xml`. Ukázka demonstruje, jak použít `presrc` vnořený prvek uvnitř úlohy Managed File Transfer fte: filemove ke spuštění skriptu Ant před provedením operace přesunu přenosu souborů. Skript `zipfiles.xml` volaný vnořeným prvkem `presrc` ve skriptu `zip.xml` komprimuje obsah adresáře. Skript `zip.xml` přenese komprimovaný soubor. Tato ukázka vyžaduje, aby byl skript `zipfiles.xml` Ant přítomen v cestě příkazu zdrojového agenta. Je to proto, že skript `zipfiles.xml` Ant obsahuje cíl použitý ke kompresi obsahu adresáře na zdrojovém agentovi. Další informace o nastavení cesty k příkazům agenta viz commandPath MFT property.

Související pojmy

“Začínáme s použitím skriptů Ant s produktem MFT” na stránce 1182
Použití skriptů Ant s produktem Managed File Transfer vám umožňuje koordinovat složité operace přenosu souborů z interpretovaného skriptovacího jazyka.

Související odkazy

fteAnt: spuštění Ant úloh v MFT

Přizpůsobení produktu MFT pomocí uživatelských procedur

Funkce produktu Managed File Transfer můžete přizpůsobit pomocí vlastních programů známých jako uživatelské procedury.

Důležité: Jakýkoli kód v rámci uživatelské procedury není produktem IBMpodporován a jakékoli problémy s tímto kódem musí být nejprve prozkoumány buď vaším podnikem, nebo dodavatelem, který poskytl uživatelskou proceduru.

Managed File Transfer poskytuje body v kódu, kde může produkt Managed File Transfer předat řízení programu, který jste napsali (uživatelská procedura). Tyto body jsou známé jako uživatelské výstupní body. Produkt Managed File Transfer pak může pokračovat v řízení po dokončení práce vašeho programu. Nemusíte používat žádné uživatelské procedury, ale jsou užitečné, pokud chcete rozšířit a upravit funkci svého systému Managed File Transfer tak, aby splňovala vaše specifické požadavky.

Existují dva body během zpracování přenosu souborů, kde můžete vyvolat uživatelskou proceduru na zdrojovém systému a dva body během zpracování přenosu souborů, kde můžete vyvolat uživatelskou proceduru na cílovém systému. Následující tabulka shrnuje každý z těchto bodů uživatelských procedur a rozhraní Java , které musíte implementovat, abyste mohli používat body předání řízení uživatelskému programu.

Tabulka 187. Souhrn bodů předání řízení uživatelskému programu na straně zdroje a na straně cíle a rozhraní Java

Bod ukončení	**Rozhraní Java , které se má implementovat**
Body předání řízení uživatelskému programu na straně zdroje:	
Před zahájením celého přenosu souborů	SourceTransferStartExitrozhraní .java
Po dokončení celého přenosu souborů	SourceTransferEndExit.java
Body předání řízení uživatelskému programu na straně cíle:	
Před zahájením celého přenosu souborů	DestinationTransferStartExitrozhraní .java
Po dokončení celého přenosu souborů	DestinationTransferEndExitrozhraní .java

Uživatelské procedury jsou vyvolány v následujícím pořadí:

1. SourceTransferStartExit
2. DestinationTransferStartExit
3. DestinationTransferEndExit
4. SourceTransferEndExit

Změny provedené uživatelskými procedurami SourceTransferStartExit a DestinationTransferStartExit jsou šířeny jako vstup do následných uživatelských procedur. Pokud například uživatelská procedura SourceTransferStartExit upraví metadata přenosu, změny se projeví v metadatech vstupního přenosu do ostatních uživatelských procedur.

Uživatelské procedury a volání programu jsou vyvolána v následujícím pořadí:

```
- SourceTransferStartExit(onSourceTransferStart).
- PRE_SOURCE Command.
- DestinationTransferStartExits(onDestinationTransferStart).
- PRE_DESTINATION Command.
- The Transfer request is performed.
- DestinationTransferEndExits(onDestinationTransferENd).
- POST_DESTINATION Command.
- SourceTransferEndExits(onSourceTransferEnd.
- POST_SOURCE Command.
```

Notes:

1. **DestinationTransferEndExits** se spustí pouze po dokončení přenosu, buď úspěšně, nebo částečně úspěšně.
2. **postDestinationCall** se spustí pouze po dokončení přenosu, buď úspěšně, nebo částečně úspěšně.
3. **SourceTransferEndExits** se spustí pro úspěšné, částečně úspěšné nebo nezdařené přenosy.

4. **postSourceCall** se volá pouze v případě, že:
   - Přenos nebyl zrušen.
   - Existuje úspěšný nebo částečně úspěšný výsledek.
   - Všechny programy přenosu po cíli byly úspěšně spuštěny.

Sestavení uživatelské procedury

Rozhraní pro sestavení uživatelské procedury jsou obsažena v souboru *MQ_INSTALL_DIRECTORY*/mqft/lib/com.ibm.wmqfte.exitroutines.api.jar. Tento soubor .jar musíte zahrnout do cesty ke třídám při sestavování uživatelské procedury. Chcete-li uživatelskou proceduru spustit, extrahujte ji jako soubor .jar a umístěte tento soubor .jar do adresáře, jak je popsáno v následující sekci.

Umístění uživatelských procedur

Uživatelské procedury můžete uložit do dvou možných umístění:

- Adresář exits. Pod každým adresářem agenta se nachází adresář uživatelských procedur. Příklad: var\mqm\mqft\config\QM_JUPITER\agents\AGENT1\exits
- Můžete nastavit vlastnost cesty exitClasstak, aby určovala alternativní umístění. Pokud jsou výstupní třídy v adresáři exits i v cestě ke třídám nastavené pomocí cesty exitClass, třídy v adresáři exits mají prioritu, což znamená, že pokud jsou třídy v obou umístěních se stejným názvem, třídy v adresáři exits mají prioritu.

Konfigurace agenta pro použití uživatelských procedur

Existují čtyři vlastnosti agenta, které lze nastavit tak, aby určovaly uživatelské procedury, které agent vyvolává. Tyto vlastnosti agenta jsou sourceTransferStartExitClasses, sourceTransferEndExitClasses, destinationTransferStartExitClassesa destinationTransferEndExitClasses. Chcete-li získat informace o tom, jak používat tyto vlastnosti, prohlédněte si téma MFT Vlastnosti agenta pro uživatelské procedury.

Spuštění uživatelských procedur na agentech mostu protokolů

Když zdrojový agent vyvolá uživatelskou proceduru, předá uživatelské proceduře seznam zdrojových položek pro přenos. Pro normální agenty se jedná o seznam úplných názvů souborů. Vzhledem k tomu, že soubory by měly být lokální (nebo přístupné prostřednictvím připojení), je uživatelská procedura schopna k nim přistupovat a šifrovat je.

Avšak pro agenta mostu protokolů jsou položky v seznamu v následujícím formátu:

```
"<file server identifier>:<fully-qualified file name of the file on the remote file server>"
```

Pro každou položku v seznamu se musí uživatelská procedura nejprve připojit k souborovému serveru (pomocí protokolu FTP). Protokoly FTPS nebo SFTP), stáhněte soubor, zašifrujte jej lokálně a pak odešlete zašifrovaný soubor zpět na souborový server.

Spuštění uživatelských procedur na agentech mostu Connect:Direct

V agentech mostu Connect:Direct nelze spouštět uživatelské procedury.

Související pojmy

“MFT zdrojové a cílové uživatelské procedury” na stránce 1188

Metadata pro uživatelské procedury MFT

Java rozhraní pro uživatelské procedury MFT

Související odkazy

“Povolení vzdáleného ladění pro uživatelské procedury MFT” na stránce 1192

Při vývoji uživatelských procedur můžete použít ladicí program, který vám pomůže vyhledat problémy ve vašem kódu.

“Ukázková uživatelská procedura přenosu zdroje MFT” na stránce 1193

“Ukázková uživatelská procedura pověření mostu protokolů” na stránce 1194

Uživatelské procedury monitoru prostředků MFT

MFT Vlastnosti agenta pro uživatelské procedury

MFT zdrojové a cílové uživatelské procedury

Oddělovače adresářů

Oddělovače adresářů ve specifikacích zdrojových souborů jsou vždy reprezentovány pomocí dopředného lomítka (/) bez ohledu na to, jak jste zadali oddělovače adresářů v příkazu **fteCreateTransfer** nebo v souboru IBM MQ Explorer. Tuto skutečnost musíte vzít v úvahu při psaní uživatelské procedury. Pokud například chcete zkontrolovat, zda existuje následující zdrojový soubor: `c:\a\b.txt` a zadali jste tento zdrojový soubor pomocí příkazu **fteCreateTransfer** nebo IBM MQ Explorer, všimněte si, že název souboru je ve skutečnosti uložen jako: `c:/a/b.txt` Takže pokud hledáte původní řetězec `c:\a\b.txt`, nenaleznete shodu.

Výstupní body na straně zdroje

Před zahájením celého přenosu souborů

Tato uživatelská procedura je volána zdrojovým agentem, když je požadavek na přenos další v seznamu nevyřízených přenosů a přenos se chystá spustit.

Příkladem použití tohoto bodu předání řízení uživatelskému programu je odeslání souborů ve fázích do adresáře, ke kterému má agent přístup pro čtení/zápis pomocí externího příkazu, nebo přejmenování souborů na cílovém systému.

Předejte této uživatelské proceduře následující argumenty:

- Název zdrojového agenta
- Název cílového agenta
- Metadata prostředí
- Metadata přenosu
- Specifikace souboru (včetně metadat souboru)

Data vrácená z této uživatelské procedury jsou následující:

- Metadata přenosu byla aktualizována. Položky lze přidávat, upravovat a odstraňovat.
- Aktualizovaný seznam specifikací souborů, který se skládá z dvojic názvu zdrojového souboru a názvu cílového souboru. Položky lze přidávat, upravovat a odstraňovat.
- Indikátor, který určuje, zda se má pokračovat v přenosu
- Řetězec, který se má vložit do protokolu přenosu.

Implementujte SourceTransferStartExit.java pro volání kódu uživatelské procedury v tomto bodě předání řízení uživatelskému programu.

Po dokončení celého přenosu souborů

Tato uživatelská procedura je volána zdrojovým agentem po dokončení celého přenosu souborů.

Příkladem použití tohoto bodu předání řízení uživatelskému programu je provedení některých úloh dokončení, jako je například odeslání e-mailu nebo zprávy IBM MQ s příznakem, že přenos byl dokončen.

Předejte této uživatelské proceduře následující argumenty:

- Výsledek ukončení přenosu
- Název zdrojového agenta
- Název cílového agenta

- Metadata prostředí
- Metadata přenosu
- Výsledky souboru

Data vrácená z této uživatelské procedury jsou následující:

- Byl aktualizován řetězec pro vložení do protokolu přenosu.

Implementujte SourceTransferEndExitrozhraní .java pro volání kódu uživatelské procedury v tomto bodu předání řízení uživatelskému programu.

Výstupní body na straně cíle

Před zahájením celého přenosu souborů

Příkladem použití tohoto bodu předání řízení uživatelskému programu je ověření oprávnění v místě určení.

Předejte této uživatelské proceduře následující argumenty:

- Název zdrojového agenta
- Název cílového agenta
- Metadata prostředí
- Metadata přenosu
- Specifikace souboru

Data vrácená z této uživatelské procedury jsou následující:

- Byla aktualizována sada názvů cílových souborů. Položky lze upravit, ale nepřidávat ani neodstraňovat.
- Indikátor, který určuje, zda se má pokračovat v přenosu
- Řetězec, který se má vložit do protokolu přenosu.

Implementujte rozhraní DestinationTransferStartExit.java pro volání kódu uživatelské procedury v tomto bodě předání řízení uživatelskému programu.

Po dokončení celého přenosu souborů

Příkladem použití této uživatelské procedury je spuštění dávkového zpracování, které používá přenesené soubory, nebo odeslání e-mailu, pokud se přenos nezdařil.

Předejte této uživatelské proceduře následující argumenty:

- Výsledek ukončení přenosu
- Název zdrojového agenta
- Název cílového agenta
- Metadata prostředí
- Metadata přenosu
- Výsledky souboru

Data vrácená z této uživatelské procedury jsou následující:

- Byl aktualizován řetězec pro vložení do protokolu přenosu.

Implementujte DestinationTransferEndExitrozhraní .java pro volání kódu uživatelské procedury v tomto bodě předání řízení uživatelskému programu.

Související pojmy

Java rozhraní pro uživatelské procedury MFT

Související odkazy

“Povolení vzdáleného ladění pro uživatelské procedury MFT” na stránce 1192

Při vývoji uživatelských procedur můžete použít ladicí program, který vám pomůže vyhledat problémy ve vašem kódu.

“Ukázková uživatelská procedura přenosu zdroje MFT” na stránce 1193

Uživatelské procedury monitoru prostředků MFT

Použití uživatelských procedur I/O přenosu MFT

Uživatelské procedury I/O přenosu systému Managed File Transfer můžete použít ke konfiguraci vlastního kódu pro provedení práce I/O základního systému souborů pro přenosy systému Managed File Transfer .

Pro přenosy MFT obvykle agent vybere jednoho z vestavěných poskytovatelů I/O pro interakci s odpovídajícími systémy souborů pro přenos. Integrovaní poskytovatelé I/O podporují následující typy systému souborů:

- Běžné systémy souborů typu UNIX-type a Windows-type
- z/OS z/OS sekvenční a dělené datové sady (pouze na systému z/OS)
- IBM i IBM i nativní soubory typu save file (pouze na systému IBM i)
- Fronty produktu IBM MQ
- Vzdálené servery protokolů FTP a SFTP (pouze pro agenty mostu protokolů)
- Vzdálené uzly Connect:Direct (pouze pro agenty mostu Connect:Direct)

V případě systémů souborů, které nejsou podporovány, nebo v případě, že vyžadujete vlastní chování I/O, můžete napsat uživatelskou proceduru I/O přenosu.

Uživatelské procedury I/O přenosu používají existující infrastrukturu pro uživatelské procedury. Avšak tyto uživatelské procedury I/O přenosu se liší od ostatních uživatelských procedur, protože k jejich funkci se přistupuje vícekrát během přenosu pro každý soubor.

Pomocí vlastnosti agenta IOExitClasses (v souboru `agent.properties`) určete, které třídy procedur I/O se mají načíst. Každou třídu ukončení oddělte čárkou, například:

```
IOExitClasses=testExits.TestExit1,testExits.testExit2
```

Rozhraní Java pro uživatelské procedury I/O přenosu jsou následující:

IOExit
Hlavní vstupní bod použitý k určení, zda je použita uživatelská procedura I/O. Tato instance je zodpovědná za vytváření instancí IOExitPath .

Pro vlastnost agenta IOExitClassesje třeba určit pouze rozhraní uživatelské procedury I/O IOExit.

IOExitPath
Představuje abstraktní rozhraní; například datový kontejner nebo zástupný znak představující sadu datových kontejnerů. Nelze vytvořit instanci třídy, která implementuje toto rozhraní. Rozhraní umožňuje, aby cesta byla prozkoumána a odvozené cesty byly vypsány. Rozhraní cesty IOExitResourcea IOExitWildcardrozšiřují IOExitPath.

IOExitChannel
Umožňuje číst data z prostředku IOExitPath nebo do něj zapisovat.

Kanál IOExitRecord
Rozšiřuje rozhraní IOExitChannel pro prostředky IOExitPath orientované na záznamy, což umožňuje čtení dat z prostředku IOExitPath nebo jejich zápis do prostředku IOExitPath v násobcích záznamů.

IOExitLock
Představuje zámek na prostředku IOExitPath pro sdílený nebo výlučný přístup.

z/OS IOExitRecordResourcePath
Rozšiřuje rozhraní cesty IOExitResourcetak, aby představovalo datový kontejner pro soubor orientovaný na záznamy, například datovou sadu z/OS . Toto rozhraní můžete použít k vyhledání dat a k vytvoření instancí kanálu IOExitRecordpro operace čtení a zápisu.

Cesta IOExitResource
Rozšiřuje rozhraní IOExitPath tak, aby představovalo datový kontejner; například soubor nebo adresář. K vyhledání dat můžete použít rozhraní. Pokud rozhraní představuje adresář, můžete pomocí metody listPaths vrátit seznam cest.

Cesta IOExitWildcard
Rozšiřuje rozhraní IOExitPath tak, aby představovalo cestu, která označuje zástupný znak. Toto rozhraní můžete použít k porovnání více cest IOExitResource.

IOExitProperties
Určuje vlastnosti, které určují, jak produkt Managed File Transfer zpracovává IOExitPath pro určité aspekty vstupu/výstupu. Například, zda použít intermediační soubory nebo zda znovu načíst prostředek od začátku, pokud je přenos restartován.

Související pojmy
“Přizpůsobení produktu MFT pomocí uživatelských procedur” na stránce 1185
Funkce produktu Managed File Transfer můžete přizpůsobit pomocí vlastních programů známých jako uživatelské procedury.

Související odkazy
Rozhraní IOExit.java
Rozhraní IOExitChannel.java
Rozhraní IOExitLock.java
Rozhraní IOExitPath.java
IOExitPropertiesrozhraní .java
Rozhraní IOExitRecordChannel.java
z/OS Rozhraní IOExitRecordResourcePath.java
Rozhraní IOExitResourcePath.java
Rozhraní IOExitWildcardPath.java
Soubor MFT`agent.properties`

IBM i Ukázka MFT v uživatelských procedur IBM i

Produkt Managed File Transfer poskytuje vzorové uživatelské procedury specifické pro produkt IBM i s vaší instalací. Ukázky jsou v adresářích *MQMFT_install_dir*`/samples/ioexit-IBMi` a *MQMFT_install_dir*`/samples/userexit-IBMi`.

com.ibm.wmqfte.exit.io.ibmi.qdls.FTEQDLSExit
Ukázková uživatelská procedura com.ibm.wmqfte.exit.io.ibmi.qdls.FTEQDLSExit přenáší soubory v systému souborů QDLS na IBM i. Po instalaci uživatelské procedury budou všechny přenosy do souborů, které začínají na `/QDLS` , automaticky používat uživatelskou proceduru.

Chcete-li nainstalovat tuto uživatelskou proceduru, postupujte takto:

1. Zkopírujte soubor `com.ibm.wmqfte.samples.ibmi.ioexits.jar` z adresáře *WMQFTE_install_dir*`/samples/ioexit-IBMi` do adresáře `exits` agenta.
2. Přidejte com.ibm.wmqfte.exit.io.ibmi.qdls.FTEQDLSExit do vlastnosti IOExitClasses .
3. Spusťte agenta znovu.

com.ibm.wmqfte.exit.user.ibmi.FileMemberMonitorExit

Ukázková uživatelská procedura com.ibm.wmqfte.exit.user.ibmi.FileMemberMonitorExit se chová jako monitor souborů MFT a automaticky přenáší členy fyzických souborů z knihovny IBM i .

Chcete-li spustit tuto uživatelskou proceduru, zadejte hodnotu pro pole metadat "library.qsys.monitor" (například pomocí parametru **-md**). Tento parametr vezme cestu ve stylu IFS ke členu souboru a může obsahovat zástupné znaky souboru a členu. Například /QSYS.LIB/FOO.LIB/BAR.FILE/*.MBR, /QSYS.LIB/FOO.LIB/*.FILE/BAR.MBR, /QSYS.LIB/FOO.LIB/*.SOUBOR/*.MBR.

Tato ukázková uživatelská procedura má také volitelné pole metadat "naming.scheme.qsys.monitor", které můžete použít k určení schématu pojmenování, které se používá během přenosu. Standardně je toto pole nastaveno na hodnotu "unix,", což způsobí, že se cílový soubor bude nazývat F00.MBR. Můžete také zadat hodnotu "ibmi" pro použití IBM i FTP FILE.MEMBER , například /QSYS.LIB/FOO.LIB/BAR.FILE/BAZ.MBR se přenáší jako BAR.BAZ.

Chcete-li nainstalovat tuto uživatelskou proceduru, postupujte takto:

1. Zkopírujte soubor `com.ibm.wmqfte.samples.ibmi.userexits.jar` z adresáře *`WMQFTE_install_dir`*`/samples/userexit-IBMi` do adresáře `exits` agenta.
2. Přidejte com.ibm.wmqfte.exit.user.ibmi.FileMemberMonitorExit do vlastnosti tříd sourceTransferStartExitv souboru `agent.properties` .
3. Spusťte agenta znovu.

com.ibm.wmqfte.exit.user.ibmi.EmptyFileDeleteExit

Ukázková uživatelská procedura com.ibm.wmqfte.exit.user.ibmi.EmptyFileDeleteExit odstraní při odstranění člena zdrojového souboru jako součást přenosu prázdný objekt souboru. Vzhledem k tomu, že objekty souboru IBM i mohou potenciálně obsahovat mnoho členů, jsou objekty souboru produktem MFTpovažovány za adresáře. Proto nemůžete provést operaci přesunu na objektu souboru pomocí MFT; operace přesunu jsou podporovány pouze na úrovni členu. V důsledku toho, když provádíte operaci přesunu na členu, nyní prázdný soubor zůstane pozadu. Tuto ukázkovou uživatelskou proceduru použijte, chcete-li tyto prázdné soubory odstranit jako součást požadavku na přenos.

Zadáte-li pro metadata "empty.file.delete" hodnotu "true" a přenesete-li FTEFileMember, ukázková uživatelská procedura odstraní nadřízený soubor, pokud je soubor prázdný.

Chcete-li nainstalovat tuto uživatelskou proceduru, postupujte takto:

1. Zkopírujte soubor com.ibm.wmqfte.samples.ibmi.userexits.jar z adresáře *`WMQFTE_install_dir`*`/samples/userexit-IBMi` do adresáře `exits` agenta.
2. Přidejte com.ibm.wmqfte.exit.user.ibmi.EmptyFileDeleteExit do vlastnosti tříd sourceTransferStartExitv souboru `agent.properties` .
3. Spusťte agenta znovu.

Související odkazy

“Použití uživatelských procedur I/O přenosu MFT” na stránce 1190
Uživatelské procedury I/O přenosu systému Managed File Transfer můžete použít ke konfiguraci vlastního kódu pro provedení práce I/O základního systému souborů pro přenosy systému Managed File Transfer .

MFT Vlastnosti agenta pro uživatelské procedury

Povolení vzdáleného ladění pro uživatelské procedury MFT

Při vývoji uživatelských procedur můžete použít ladicí program, který vám pomůže vyhledat problémy ve vašem kódu.

Protože uživatelské procedury běží uvnitř virtuálního počítače Java , který spouští agenta, nemůžete použít podporu přímého ladění, která je obvykle zahrnuta v integrovaném vývojovém prostředí. Můžete však povolit vzdálené ladění prostředí JVM a poté připojit vhodný vzdálený ladicí program.

Chcete-li povolit vzdálené ladění, použijte standardní parametry prostředí JVM **-Xdebug** a **-Xrunjdwp**. Tyto vlastnosti jsou předány prostředí JVM, které spouští agenta, pomocí proměnné prostředí **BFG_JVM_PROPERTIES** . Například v systému AIX and Linux následující příkazy spustí agenta a způsobí, že prostředí JVM bude naslouchat připojením ladicího programu na portu TCP 8765.

```
export BFG_JVM_PROPERTIES="-Xdebug -Xrunjdwp:transport=dt_socket,server=y,address=8765"
fteStartAgent -F TEST_AGENT
```

Agent se nespustí, dokud se ladicí program nepřipojí. Použijte příkaz **set** na systému Windows místo příkazu **export** .

Můžete také použít jiné komunikační metody mezi ladicím programem a prostředím JVM. Prostředí JVM může například místo naopak otevřít připojení k ladicímu programu, nebo můžete místo protokolu TCP použít sdílenou paměť. Další podrobnosti viz dokumentace Java Architektura ladicího programu platformy .

Musíte použít parametr **-F** (popředí), když spouštíte agenta ve vzdáleném režimu ladění.

Použití ladicího programu Eclipse

Následující kroky platí pro schopnost vzdáleného ladění ve vývojovém prostředí Eclipse . Můžete také použít jiné vzdálené ladicí programy, které jsou kompatibilní s JPDA.

1. Klepněte na volbu **Spustit** > **Otevřít dialogové okno Ladění** (nebo **Spustit** > **Konfigurace ladění** nebo **Spustit** > **Dialogové okno Ladění** v závislosti na vaší verzi produktu Eclipse).
2. Poklepáním na volbu **Vzdálená aplikace Java** v seznamu typů konfigurace vytvořte konfiguraci ladění.
3. Vyplňte pole konfigurace a uložte konfiguraci ladění. Pokud jste již spustili prostředí JVM agenta v režimu ladění, můžete se nyní připojit k prostředí JVM.

Ukázková uživatelská procedura přenosu zdroje MFT

```
/*
 * A Sample Source Transfer End Exit that prints information about a transfer to standard
output.
 * If the agent is run in the background the output will be sent to the agent's event log file.
If
 * the agent is started in the foreground by specifying the -F parameter on the fteStartAgent
 * command the output will be sent to the console.
 *
 * To run the exit execute the following steps:
 *
 * Compile and build the exit into a jar file. You need the following in the class path:
 * {MQ_INSTALLATION_PATH}\mqft\lib\com.ibm.wmqfte.exitroutines.api.jar
 *
 * Put the jar in your agent's exits directory:
 * {MQ_DATA_PATH}\config\coordQmgrName\agents\agentName\exits\
 *
 * Update the agent's properties file:
 * {MQ_DATA_PATH}\config\coordQmgrName\agents\agentName\agent.properties
 * to include the following property:
 * sourceTransferEndExitClasses=[packageName.]SampleEndExit
 *
 * Restart agent to pick up the exit
 *
 * Send the agent a transfer request:
 * For example: fteCreateTransfer -sa myAgent -da YourAgent -df output.txt input.txt
 */

import java.util.List;
import java.util.Map;
import java.util.Iterator;

import com.ibm.wmqfte.exitroutine.api.SourceTransferEndExit;
import com.ibm.wmqfte.exitroutine.api.TransferExitResult;
import com.ibm.wmqfte.exitroutine.api.FileTransferResult;

public class SampleEndExit implements SourceTransferEndExit {
```

```
    public String onSourceTransferEnd(TransferExitResult transferExitResult,
            String sourceAgentName,
            String destinationAgentName,
            Map<String, String>environmentMetaData,
            Map<String, String>transferMetaData,
            List<FileTransferResult>fileResults) {

        System.out.println("Environment Meta Data: " + environmentMetaData);
        System.out.println("Transfer Meta Data:    " + transferMetaData);

        System.out.println("Source agent:          " +
                sourceAgentName);
        System.out.println("Destination agent:     " +
                destinationAgentName);

        if (fileResults.isEmpty()) {
            System.out.println("No files in the list");
            return "No files";
        }
        else {

            System.out.println( "File list: ");

            final Iterator<FileTransferResult> iterator = fileResults.iterator();

            while (iterator.hasNext()){
                final FileTransferResult thisFileSpec = iterator.next();
                System.out.println("Source file spec: "  +
                        thisFileSpec.getSourceFileSpecification() +
                        ", Destination file spec: " +
                        thisFileSpec.getDestinationFileSpecification());
            }
        }
        return "Done";
    }
}
```

Ukázková uživatelská procedura pověření mostu protokolů

Informace o použití této ukázkové uživatelské procedury naleznete v tématu Mapování pověření pro souborový server pomocí tříd ukončení.

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.Map;
import java.util.Properties;
import java.util.StringTokenizer;

import com.ibm.wmqfte.exitroutine.api.CredentialExitResult;
import com.ibm.wmqfte.exitroutine.api.CredentialExitResultCode;
import com.ibm.wmqfte.exitroutine.api.CredentialPassword;
import com.ibm.wmqfte.exitroutine.api.CredentialUserId;
import com.ibm.wmqfte.exitroutine.api.Credentials;
import com.ibm.wmqfte.exitroutine.api.ProtocolBridgeCredentialExit;

/**
 * A sample protocol bridge credential exit
 *
 * This exit reads a properties file that maps mq user ids to server user ids
 * and server passwords. The format of each entry in the properties file is:
 *
 * mqUserId=serverUserId,serverPassword
 *
 * The location of the properties file is taken from the protocol bridge agent
 * property protocolBridgeCredentialConfiguration.
 *
 * To install the sample exit compile the class and export to a jar file.
 * Place the jar file in the exits subdirectory of the agent data directory
 * of the protocol bridge agent on which the exit is to be installed.
 * In the agent.properties file of the protocol bridge agent set the
```

```
 * protocolBridgeCredentialExitClasses to SampleCredentialExit
 * Create a properties file that contains the mqUserId to serverUserId and
 * serverPassword mappings applicable to the agent. In the agent.properties
 * file of the protocol bridge agent set the protocolBridgeCredentialConfiguration
 * property to the absolute path name of this properties file.
 * To activate the changes stop and restart the protocol bridge agent.
 *
 * For further information on protocol bridge credential exits refer to
 * the WebSphere MQ Managed File Transfer documentation online at:
 * https://www.ibm.com/docs/SSEP7X_7.0.4/welcome/WelcomePagev7r0.html
 */
public class SampleCredentialExit implements ProtocolBridgeCredentialExit {

    // The map that holds mq user ID to serverUserId and serverPassword mappings
    final private Map<String,Credentials> credentialsMap = new HashMap<String, Credentials>();

    /* (non-Javadoc)
      * @see com.ibm.wmqfte.exitroutine.api.ProtocolBridgeCredentialExit#initialize(java.util.Map)
      */
     public synchronized boolean initialize(Map<String, String> bridgeProperties) {

        // Flag to indicate whether the exit has been successfully initialized or not
        boolean initialisationResult = true;

        // Get the path of the mq user ID mapping properties file
         final String propertiesFilePath = bridgeProperties.get("protocolBridgeCredentialConfiguration");

         if (propertiesFilePath == null || propertiesFilePath.length() == 0) {
             // The properties file path has not been specified. Output an error and return false
             System.err.println("Error initializing SampleCredentialExit.");
              System.err.println("The location of the mqUserID mapping properties file has not been
specified in the
       protocolBridgeCredentialConfiguration property");
             initialisationResult = false;
         }

        if (initialisationResult) {

            // The Properties object that holds mq user ID to serverUserId and serverPassword
             // mappings from the properties file
             final Properties mappingProperties = new Properties();

             // Open and load the properties from the properties file
             final File propertiesFile = new File (propertiesFilePath);
             FileInputStream inputStream = null;
             try {
                 // Create a file input stream to the file
                 inputStream = new FileInputStream(propertiesFile);

                 // Load the properties from the file
                 mappingProperties.load(inputStream);
             }
             catch (FileNotFoundException ex) {
                 System.err.println("Error initializing SampleCredentialExit.");
                  System.err.println("Unable to find the mqUserId mapping properties file: " +
propertiesFilePath);
                 initialisationResult = false;
             }
             catch (IOException ex) {
                 System.err.println("Error initializing SampleCredentialExit.");
                  System.err.println("Error loading the properties from the mqUserId mapping properties
file: " + propertiesFilePath);
                 initialisationResult = false;
             }
             finally {
                 // Close the inputStream
                 if (inputStream != null) {
                     try {
                         inputStream.close();
                     }
                     catch (IOException ex) {
                         System.err.println("Error initializing SampleCredentialExit.");
                          System.err.println("Error closing the mqUserId mapping properties file: " +
propertiesFilePath);
                         initialisationResult = false;
                     }
                 }
             }

             if (initialisationResult) {
                 // Populate the map of mqUserId to server credentials from the properties
                 final Enumeration<?> propertyNames = mappingProperties.propertyNames();
```

```
                  while ( propertyNames.hasMoreElements()) {
                     final Object name = propertyNames.nextElement();
                     if (name instanceof String ) {
                         final String mqUserId = ((String)name).trim();
                          // Get the value and split into serverUserId and serverPassword
                        final String value = mappingProperties.getProperty(mqUserId);
                        final StringTokenizer valueTokenizer = new StringTokenizer(value, ",");
                         String serverUserId = "";
                         String serverPassword = "";
                         if (valueTokenizer.hasMoreTokens()) {
                             serverUserId = valueTokenizer.nextToken().trim();
                         }
                         if (valueTokenizer.hasMoreTokens()) {
                             serverPassword = valueTokenizer.nextToken().trim();
                         }
                         // Create a Credential object from the serverUserId and serverPassword
final Credentials credentials = new Credentials(new CredentialUserId(serverUserId), new
CredentialPassword(serverPassword));
                         // Insert the credentials into the map
                         credentialsMap.put(mqUserId, credentials);
                     }
                 }

            }
         }

         return initialisationResult;
     }
      /* (non-Javadoc)
      * @see com.ibm.wmqfte.exitroutine.api.ProtocolBridgeCredentialExit#mapMQUserId(java.lang.String)
      */
     public synchronized CredentialExitResult mapMQUserId(String mqUserId) {
         CredentialExitResult result = null;
         // Attempt to get the server credentials for the given mq user id
         final Credentials credentials = credentialsMap.get(mqUserId.trim());
         if ( credentials == null) {
             // No entry has been found so return no mapping found with no credentials
             result = new CredentialExitResult(CredentialExitResultCode.NO_MAPPING_FOUND, null);
         }
         else {
             // Some credentials have been found so return success to the user along with the credentials
             result = new CredentialExitResult(CredentialExitResultCode.USER_SUCCESSFULLY_MAPPED,
credentials);
         }
         return result;
     }
      /* (non-Javadoc)
      * @see com.ibm.wmqfte.exitroutine.api.ProtocolBridgeCredentialExit#shutdown(java.util.Map)
      */
     public void shutdown(Map<String, String> bridgeProperties) {
         // Nothing to do in this method because there are no resources that need to be released
     }
 }
```

Ukázková uživatelská procedura vlastností mostu protokolů

Informace o použití této ukázkové uživatelské procedury naleznete v tématu ProtocolBridgePropertiesExit2: Vyhledávání vlastností souborového serveru protokolu

SamplePropertiesExit2.java

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import java.util.Map.Entry;
import java.util.Properties;

import com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit2;
import com.ibm.wmqfte.exitroutine.api.ProtocolServerPropertyConstants;

/**
 * A sample protocol bridge properties exit. This exit reads a properties file
 * that contains properties for protocol servers.
```

```
 * <p>
 * The format of each entry in the properties file is:
 * {@literal serverName=type://host:port}
 * Ensure there is a default entry such as
 * {@literal default=type://host:port}
 * otherwise the agent will fail to start with a BFGBR0168 as it must have a
 * default server.
 * <p>
 * The location of the properties file is taken from the protocol bridge agent
 * property {@code protocolBridgePropertiesConfiguration}.
 * <p>
 * The methods {@code getCredentialLocation} returns the location of the associated
 * ProtocolBridgeCredentials.xml, this sample it is defined to be stored in a directory
 * defined by the environment variable CREDENTIALSHOME
 * <p>
 * To install the sample exit:
 * <ol>
 * <li>Compile the class and export to a jar file.
 * <li>Place the jar file in the {@code exits} subdirectory of the agent data directory
 * of the protocol bridge agent on which the exit is to be installed.
 * <li>In the {@code agent.properties} file of the protocol bridge agent
 * set the {@code protocolBridgePropertiesExitClasses} to
 * {@code SamplePropertiesExit2}.
 * <li>Create a properties file that contains the appropriate properties to specify the
 * required servers.
 * <li>In the {@code agent.properties} file of the protocol bridge agent
 * set the <code>protocolBridgePropertiesConfiguration</code> property to the
 * absolute path name of this properties file.
 * <li>To activate the changes stop and restart the protocol bridge agent.
 * </ol>
 * <p>
 * For further information on protocol bridge properties exits refer to the
 * WebSphere MQ Managed File Transfer documentation online at:
 * <p>
 * {@link https://www.ibm.com/docs/SSEP7X_7.0.4/welcome/WelcomePagev7r0.html}
 */
public class SamplePropertiesExit2 implements ProtocolBridgePropertiesExit2 {

    /**
     * Helper class to encapsulate protocol server information.
     */
    private static class ServerInformation {
        private final String type;
        private final String host;
        private final int port;

        public ServerInformation(String url) {
            int index = url.indexOf("://");
            if (index == -1) throw new IllegalArgumentException("Invalid server URL: "+url);
            type = url.substring(0, index);

            int portIndex = url.indexOf(":", index+3);
            if (portIndex == -1) {
                host = url.substring(index+3);
                port = -1;
            } else {
                host = url.substring(index+3,portIndex);
                port = Integer.parseInt(url.substring(portIndex+1));
            }
        }

        public String getType() {
            return type;
        }

        public String getHost() {
            return host;
        }

        public int getPort() {
            return port;
        }
    }

    /** A {@code Map} that holds information for each configured protocol server */
    final private Map<String, ServerInformation> servers = new HashMap<String, ServerInformation>();


    /* (non-Javadoc)
     * @see
com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit#getProtocolServerProperties(java.lang.String)
     */
```

```
    public Properties getProtocolServerProperties(String protocolServerName) {
        // Attempt to get the protocol server information for the given protocol server name
        // If no name has been supplied then this implies the default.
        final ServerInformation info;
        if (protocolServerName == null || protocolServerName.length() == 0) {
            protocolServerName = "default";
        }
        info = servers.get(protocolServerName);

        // Build the return set of properties from the collected protocol server information, when
available.
        // The properties set here is the minimal set of properties to be a valid set.
        final Properties result;
        if (info != null) {
            result = new Properties();
            result.setProperty(ProtocolServerPropertyConstants.SERVER_NAME, protocolServerName);
            result.setProperty(ProtocolServerPropertyConstants.SERVER_TYPE, info.getType());
            result.setProperty(ProtocolServerPropertyConstants.SERVER_HOST_NAME, info.getHost());
            if (info.getPort() != -1)
result.setProperty(ProtocolServerPropertyConstants.SERVER_PORT_VALUE, ""+info.getPort());
            result.setProperty(ProtocolServerPropertyConstants.SERVER_PLATFORM, "UNIX");
            if (info.getType().toUpperCase().startsWith("FTP")) {        // FTP & FTPS
                result.setProperty(ProtocolServerPropertyConstants.SERVER_TIMEZONE, "Europe/London");
                result.setProperty(ProtocolServerPropertyConstants.SERVER_LOCALE, "en-GB");
            }
            result.setProperty(ProtocolServerPropertyConstants.SERVER_FILE_ENCODING, "UTF-8");
        } else {
            System.err.println("Error no default protocol file server entry has been supplied");
            result = null;
        }

        return result;
    }

    /* (non-Javadoc)
     * @see com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit#initialize(java.util.Map)
     */
    public boolean initialize(Map<String, String> bridgeProperties) {
        // Flag to indicate whether the exit has been successfully initialized or not
        boolean initialisationResult = true;

        // Get the path of the properties file
        final String propertiesFilePath = bridgeProperties.get("protocolBridgePropertiesConfiguration");
        if (propertiesFilePath == null || propertiesFilePath.length() == 0) {
            // The protocol server properties file path has not been specified. Output an error and
return false
            System.err.println("Error initializing SamplePropertiesExit.");
            System.err.println("The location of the protocol server properties file has not been
specified in the
      protocolBridgePropertiesConfiguration property");
            initialisationResult = false;
        }

        if (initialisationResult) {
            // The Properties object that holds protocol server information
            final Properties mappingProperties = new Properties();

            // Open and load the properties from the properties file
            final File propertiesFile = new File (propertiesFilePath);
            FileInputStream inputStream = null;
            try {
                // Create a file input stream to the file
                inputStream = new FileInputStream(propertiesFile);

                // Load the properties from the file
                mappingProperties.load(inputStream);
            } catch (final FileNotFoundException ex) {
                System.err.println("Error initializing SamplePropertiesExit.");
                System.err.println("Unable to find the protocol server properties file: " +
propertiesFilePath);
                initialisationResult = false;
            } catch (final IOException ex) {
                System.err.println("Error initializing SamplePropertiesExit.");
                System.err.println("Error loading the properties from the protocol server properties
file: " + propertiesFilePath);
                initialisationResult = false;
            } finally {
                // Close the inputStream
                if (inputStream != null) {
                    try {
                        inputStream.close();
                    } catch (final IOException ex) {
```

```
                        System.err.println("Error initializing SamplePropertiesExit.");
                        System.err.println("Error closing the protocol server properties file: " +
propertiesFilePath);
                        initialisationResult = false;
                    }
                }
            }

            if (initialisationResult) {
                // Populate the map of protocol servers from the properties
                for (Entry<Object, Object> entry : mappingProperties.entrySet()) {
                    final String serverName = (String)entry.getKey();
                    final ServerInformation info = new ServerInformation((String)entry.getValue());
                    servers.put(serverName, info);
                }
            }
        }

        return initialisationResult;
    }

    /* (non-Javadoc)
     * @see com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit#shutdown(java.util.Map)
     */
    public void shutdown(Map<String, String> bridgeProperties) {
        // Nothing to do in this method because there are no resources that need to be released
    }

    /* (non-Javadoc)
     * @see com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit2#getCredentialLocation()
     */
    public String getCredentialLocation() {
        String envLocationPath;
        if (System.getProperty("os.name").toLowerCase().contains("win")) {
            // Windows style
            envLocationPath = "%CREDENTIALSHOME%\\ProtocolBridgeCredentials.xml";
        }
        else {
            // Unix style
            envLocationPath = "$CREDENTIALSHOME/ProtocolBridgeCredentials.xml";
        }
        return envLocationPath;
    }

}
```

Řízení MFT vložením zpráv do fronty příkazů agenta

Aplikaci, která řídí produkt Managed File Transfer , můžete napsat vložením zpráv do front příkazů agenta.

Můžete vložit zprávu do fronty příkazů agenta, chcete-li požadovat, aby agent provedl jednu z následujících akcí:

- Vytvořit přenos souborů
- Vytvořit naplánovaný přenos souborů
- Zrušit přenos souborů
- Zrušit naplánovaný přenos souborů
- Volání příkazu
- Vytvořit monitor
- Odstranit monitor
- Vrátit příkaz ping, který označuje, že je agent aktivní

Chcete-li požadovat, aby agent provedl jednu z těchto akcí, musí být zpráva ve formátu XML, který vyhovuje jednomu z následujících schémat:

FileTransfer.xsd
: Zprávy v tomto formátu lze použít k vytvoření přenosu souborů nebo plánovaného přenosu souborů, k volání příkazu nebo ke zrušení přenosu souborů nebo plánovaného přenosu souborů. Další informace viz Formát zprávy požadavku na přenos souborů.

Monitor.xsd
Zprávy v tomto formátu lze použít k vytvoření nebo odstranění monitoru prostředků. Další informace naleznete v tématu Formáty zpráv požadavků monitoru MFT.

PingAgent.xsd
Zprávy v tomto formátu lze použít k testování spojení s agentem, aby se ověřilo, že je aktivní. Další informace viz Ping MFT agent request message format.

Agent vrátí odpověď na zprávy požadavku. Zpráva odpovědi je vložena do fronty odpovědí, která je definována ve zprávě požadavku. Zpráva odpovědi je ve formátu XML definovaném následujícím schématem:

Reply.xsd
Další informace viz Formát zprávy odpovědi agenta MFT.

Vyvíjení aplikací pro MQ Telemetry

Aplikace telemetrie integrují smyslová a kontrolní zařízení s dalšími zdroji informací, které jsou k dispozici na internetu a v podnicích.

Vyvíjejte aplikace pro systém MQ Telemetry pomocí návrhových vzorů, opracovaných příkladů, ukázkových programů, konceptů programování a referenčních informací.

Související pojmy

MQ Telemetry

Případy použití telemetrie

Související úlohy

Instalace produktu MQ Telemetry

Správa serveruMQ Telemetry

Odstraňování problémů s produktem MQ Telemetry

Související odkazy

Odkaz na MQ Telemetry

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 produkt IBM MQ 8.0.0 a novější již není ukázková klientská aplikace zahrnuta v produktu MQ Telemetry. Byla součástí souboru IBM Messaging Telemetry Clients SupportPac(již není k dispozici). 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 jeho obsah:
    - Byl poskytnut pouze zdroj ukázkové aplikace (`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) produktu IBM Messaging Telemetry Clients SupportPac, informace o instalaci a spuštění ukázkové aplikace najdete v části Ověření instalace produktu MQ Telemetry pomocí příkazového řádku.

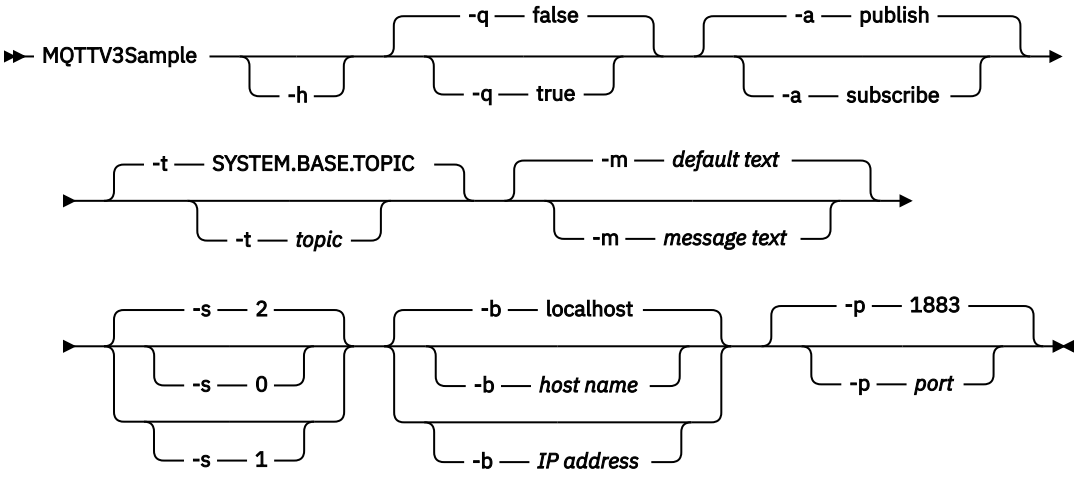
Program MQTTV3Sample

Referenční informace o ukázkové syntaxi a parametrech pro program `MQTTV3Sample` .

Účel

Program `MQTTV3Sample` lze použít k publikování zprávy a k odběru tématu. Informace o tom, jak získat tento ukázkový program, viz “IBM MQ Telemetry Transport ukázkové programy” na stránce 1200.

MQTTV3Sample syntax



Parametry

-h
Vytisknout tento text nápovědy a ukončit

-q
Nastavte tichý režim namísto použití výchozího režimu false.

-a
Místo předpokladu výchozí akce publikování nastavte publikování nebo odběr.

-t
Publikovat nebo odebírat téma namísto publikování nebo přihlášení k odběru výchozího tématu

-m
Publikujte text zprávy namísto odeslání výchozího textu publikace "Ahoj z aplikace MQTT v3 ".

-s
Nastavte QoS místo použití výchozí hodnoty QoS, 2.

-b
Připojte se k tomuto názvu hostitele nebo adrese IP místo připojení k výchozímu názvu hostitele, localhost.

-p
Místo výchozího portu 1883 použijte tento port.

Spusťte program MQTTV3Sample

Chcete-li se přihlásit k odběru tématu v systému Windows, použijte příkaz:

```
run MQTTV3Sample -a subscribe
```

Chcete-li publikovat zprávu v systému Windows, použijte příkaz:

```
run MQTTV3Sample
```

Koncepty programování klienta MQTT

Koncepce popsané v této části vám pomohou porozumět klientským knihovnám pro produkt MQTT protocol. Koncepce doplňují dokumentaci k rozhraní API připojeným ke klientským knihovnám.

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 jeho obsah:
    - Byl poskytnut pouze zdroj ukázkové aplikace (`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 vyvíjet a spouštět klienta MQTT , musíte tyto prostředky zkopírovat nebo nainstalovat na klientské zařízení. Není třeba instalovat samostatné běhové prostředí klienta.

Podmínky licencování pro klienty jsou přidruženy k serveru, ke kterému připojujete klienty.

Knihovny klienta MQTT jsou referenční implementace MQTT protocol. Své vlastní klienty můžete implementovat v různých jazycích vhodných pro různé platformy zařízení. Viz IBM MQ Telemetry Transport formát a protokol.

Dokumentace k rozhraní API nevytváří žádné předpoklady o tom, ke kterému serveru MQTT je klient připojen. Chování klienta se může mírně lišit při připojení k různým serverům. Následující popisy popisují chování klienta při připojení ke službě telemetrie IBM MQ .

Zpětná volání a synchronizace v klientských aplikacích MQTT

Programovací model klienta MQTT používá podprocesy ve velké míře. Podprocesy oddělují klientskou aplikaci systému MQTT v co největší míře od prodlev při přenosu zpráv na server a ze serveru. Publikování, tokeny doručení a události ztracených připojení jsou doručeny metodám ve třídě zpětného volání, která implementuje `MqttCallback`.

Zpětná volání

Poznámka: Nejnovější změny v souboru `MqttCallback`viz web Eclipse Paho . Například `MqttCallback` je definováno jako rozhraní ve verzi Paho klienta a asynchronní metody jsou poskytovány třídou Paho `MqttAsyncClient` .

Rozhraní `MqttCallback` má tři metody zpětného volání:

`connectionLost(java.lang.Throwable cause)`

Funkce `connectionLost` je volána, když chyba komunikace vede ke zrušení připojení. Také se volá, pokud server zruší připojení v důsledku chyby na serveru po navázání připojení. Chyby serveru jsou protokolovány do protokolu chyb správce front. Server zruší připojení ke klientovi a klient zavolá `MqttCallback.connectionLost`.

Jedinými vzdálenými chybami, které byly vygenerovány jako výjimky ve stejném podprocesu jako aplikace klienta, jsou výjimky z `MqttClient.connect`. Chyby zjištěné serverem po navázání připojení jsou ohlášeny zpět metodě zpětného volání `MqttCallback.connectionLost` jako `throwables`.

Typické chyby serveru, které vedou k `connectionLost` , jsou chyby autorizace. Server telemetrie se například pokusí publikovat v tématu jménem klienta, který není autorizován publikovat v tématu. Cokoli, co má za následek vrácení kódu podmínky `MQCC_FAIL` na server telemetrie, může vést ke zrušení připojení.

`deliveryComplete(IMqttDeliveryToken token)`

Klient produktu MQTT volá funkci `deliveryComplete` , aby předal token doručení zpět aplikaci klienta. Viz “Tokeny doručení” na stránce 1209. Pomocí tokenu doručení může zpětné volání přistupovat k publikované zprávě pomocí metody `token.getMessage`.

Když zpětné volání aplikace vrátí řízení klientovi MQTT po volání metodou `deliveryComplete` , doručení se dokončí. Dokud není doručení dokončeno, zprávy s `QoS` 1 nebo 2 jsou uchovány třídou perzistence.

Volání `deliveryComplete` je synchronizační bod mezi aplikací a třídou perzistence. Metoda `deliveryComplete` není nikdy volána dvakrát pro stejnou zprávu.

Když se zpětné volání aplikace vrátí z `deliveryComplete` na klienta MQTT , klient volá `MqttClientPersistence.remove` pro zprávy s `QoS` 1 nebo 2. Produkt `MqttClientPersistence.remove` odstraní lokálně uloženou kopii publikované zprávy.

Z perspektivy zpracování transakcí je volání `deliveryComplete` jednofázovou transakcí, která potvrzuje doručení. Pokud se zpracování během zpětného volání nezdaří, je při restartu klienta `MqttClientPersistence.remove` voláno znovu, aby se odstranila lokální kopie publikované zprávy. Zpětné volání není znovu voláno. Pokud používáte zpětné volání k uložení protokolu doručených zpráv, nemůžete synchronizovat protokol s klientem MQTT . Chcete-li protokol spolehlivě uložit, aktualizujte protokol ve třídě `MqttClientPersistence` .

Na token doručení a zprávu odkazuje hlavní podproces aplikace a klient MQTT . Klient MQTT provede při dokončení doručení odkaz na objekt `MqttMessage` a objekt tokenu doručení při odpojení klienta. Objekt `MqttMessage` může být uvolněn z paměti po dokončení doručení, pokud jej klientská aplikace nepřečte. Token doručení může být po odpojení relace uvolněn z paměti.

Po publikování zprávy můžete získat atributy `IMqttDeliveryToken` a `MqttMessage` . Pokusíte-li se nastavit jakékoli atributy `MqttMessage` po publikování zprávy, výsledek nebude definován.

Klient MQTT pokračuje ve zpracování potvrzení doručení, pokud se klient znovu připojí k předchozí relaci se stejným `ClientIdentifier` ; viz “Vyčistit relace” na stránce 1206. Aplikace klienta MQTT musí pro předchozí relaci nastavit hodnotu `MqttClient.CleanSession` na `false` a v nové relaci ji nastavit na hodnotu `false` . Klient MQTT vytvoří nové tokeny doručení a objekty zpráv v nové relaci pro nevyřízená doručení. Obnoví objekty pomocí třídy `MqttClientPersistence` . Pokud má aplikační klient stále odkazy na staré tokeny doručení a zprávy, dereference je. Zpětné volání aplikace je voláno v nové relaci pro všechna doručení zahájená v předchozí relaci a dokončená v této relaci.

Zpětné volání aplikace je voláno po připojení aplikačního klienta, když je dokončeno nevyřízené doručení. Před tím, než se aplikační klient připojí, může načíst nevyřízená doručení pomocí metody `MqttClient.getPendingDeliveryTokens` .

Všimněte si, že aplikace klienta původně vytvořila publikovaný objekt zprávy a jeho bajtové pole informačního obsahu. Klient MQTT na tyto objekty odkazuje. Objekt zprávy vrácený tokenem doručení v metodě `token.getMessage` nemusí být nutně stejný objekt zprávy vytvořený klientem. Pokud nová instance klienta MQTT znovu vytvoří token doručení, třída `MqttClientPersistence` znovu vytvoří objekt `MqttMessage` . Pro konzistenci `token.getMessage` vrací `null` pokud `token.isCompleted` je `true`, bez ohledu na to, zda byl objekt zprávy vytvořen aplikačním klientem nebo třídou `MqttClientPersistence` .

`messageArrived(String topic, MqttMessage message)`

Funkce `messageArrived` je volána, když je pro klienta, který odpovídá tématu odběru, doručeno publikování. `topic` je téma publikování, nikoli filtr odběrů. Tyto dva mohou být odlišné, pokud filtr obsahuje zástupné znaky.

Pokud se téma shoduje s více odběry vytvořenými klientem, klient obdrží více kopií publikování. Pokud klient publikuje do tématu, k jehož odběru se také přihlásí, obdrží kopii své vlastní publikace.

Pokud je zpráva odeslána s `QoS` 1 nebo 2, je zpráva uložena třídou `MqttClientPersistence` před MQTT voláním klienta `messageArrived`. `messageArrived` se chová jako `deliveryComplete`: volá se pouze jednou pro publikaci a lokální kopii publikace odebere produkt `MqttClientPersistence.remove` , když se produkt `messageArrived` vrátí do klienta MQTT . Klient MQTT zruší své odkazy na téma a zprávu, když se produkt `messageArrived` vrátí do klienta MQTT . Téma a objekty zpráv jsou uvolněny z paměti, pokud aplikační klient nezadržel odkaz na objekty.

Zpětná volání, podprocesy a synchronizace klientské aplikace

Klient MQTT volá metodu zpětného volání v samostatném podprocesu do hlavního podprocesu aplikace. Aplikace klienta nevytváří podproces pro zpětné volání, je vytvořena klientem MQTT .

Klient MQTT synchronizuje metody zpětného volání. V daném okamžiku je spuštěna pouze jedna instance metody zpětného volání. Synchronizace usnadňuje aktualizaci objektu, který se shoduje s tím, které publikace byly doručeny. Jedna instance produktu `MqttCallback.deliveryComplete` je spuštěna v daném okamžiku, a proto je bezpečné aktualizovat záznam tally bez další synchronizace. Je také pravda, že v daném okamžiku přichází pouze jedna publikace. Váš kód v metodě `messageArrived` může aktualizovat objekt bez jeho synchronizace. Pokud odkazujete na záznam tally nebo objekt, který se aktualizuje, v jiném podprocesu synchronizujte záznam tally nebo objekt.

Token doručení poskytuje mechanismus synchronizace mezi hlavním podprocesem aplikace a doručením publikace. Metoda `token.waitForCompletion` čeká na dokončení doručení specifické publikace nebo na vypršení volitelného časového limitu. Produkt `token.waitForCompletion` můžete použít následujícím způsobem ke zpracování jedné publikace v daném okamžiku.

Chcete-li provést synchronizaci s metodou `MqttCallback.deliveryComplete` . Pouze v případě, že se funkce `MqttCallback.deliveryComplete` vrátí do pole MQTT Klient provede `token.waitForCompletion` obnovení. Pomocí tohoto mechanizmu můžete synchronizovat spuštěný kód v produktu `MqttCallback.deliveryComplete` před spuštěním kódu v hlavním podprocesu aplikace.

Co kdybyste chtěli publikovat bez čekání na doručení každé publikace, ale chcete potvrzení o doručení všech publikací? Pokud publikujete v jednom podprocesu, poslední publikace, která se má odeslat, je také poslední, která se má doručit.

Synchronizace požadavků odeslaných na server

Tabulka 188 na stránce 1205 popisuje metody v klientu MQTT Java , které odesílají požadavek na server. Pokud aplikační klient nenastaví časový limit na dobu neurčitou, klient na server nikdy nečeká na dobu neurčitou. Pokud se klient zablokuje, jedná se buď o problém s programováním aplikace, nebo o defekt v klientu MQTT .

Tabulka 188. 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 vygeneruje výjimku.
`MqttClient.Disconnect`	Čeká na dokončení práce klienta MQTT , kterou musí provést, a na odpojení relace TCP/IP.	
`MqttClient.Subscribe`	Čeká na dokončení metody Odebírat nebo UnSubscribe .	
`MqttClient.UnSubscribe`		
`MqttClient.Publish`	Po předání požadavku klientovi MQTT se okamžitě vrátí do podprocesu aplikace.	Není.
`IMqttDeliveryToken.waitFor Completion`	Čeká na vrácení tokenu doručení.	Neomezeně, nebo jak je nastaveno jako parametr.

Související pojmy

Vyčistit relace
Klient MQTT a služba telemetrie (MQXR) udržují informace o stavu relace. Informace o stavu se používají k zajištění "alespoň jednou" a "přesně jednou" doručení a "přesně jednou" přijetí publikování. 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 volby `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 můžete použít libovolný identifikační řetězec. Je důležité mít proceduru pro přidělení identifikátorů klientů a prostředky pro konfiguraci klienta s jeho zvoleným identifikátorem.

Tokeny doručení

Poslední vůle a vyhlášení závěti
Pokud připojení klienta MQTT neočekávaně skončí, můžete nakonfigurovat produkt MQ Telemetry tak, aby odeslal publikaci "last will and testament". Předem definujte obsah publikace a téma, na které má být publikace odeslána. "Poslední vůle a závěť" je vlastnost spojení. Vytvořte jej před připojením klienta.

Perzistence zpráv v klientech MQTT
Zprávy publikování jsou trvalé, pokud jsou odeslány s kvalitou služby "alespoň jednou"nebo "přesně jednou". Můžete implementovat vlastní mechanizmus perzistence na klientovi nebo použít výchozí mechanizmus perzistence, který je poskytován s klientem. Perzistence funguje v obou směrech pro publikace odeslané klientovi nebo z klienta.

Publikace
Publikace jsou instance produktu `MqttMessage` , které jsou přidruženy k řetězci tématu. Klienti systému MQTT mohou vytvářet publikování, která se odesílají do produktu IBM MQ, a odebírat témata týkající se produktu IBM MQ , která přijímají publikování.

Kvality služby poskytované klientem MQTT
Klient systému MQTT poskytuje tři kvality služeb pro doručování publikací produktu IBM MQ a klientu MQTT : "nejvýše jednou", "alespoň jednou" a "přesně jednou". Když klient MQTT odešle požadavek na vytvoření odběru do produktu IBM MQ , odešle se požadavek s kvalitou služby "alespoň jednou".

Zachovaná publikování a klienti MQTT

Téma může mít jedno a pouze jedno zachované publikování. Pokud vytvoříte odběr tématu, které má zachované publikování, publikování vám bude okamžitě předáno.

Odběry
Vytvořte odběry pro registraci zájmu o témata publikování pomocí filtru témat. Klient může vytvořit více odběrů nebo odběr obsahující filtr témat, který používá zástupné znaky k registraci zájmu o více témat. Publikace o tématech odpovídajících filtrům jsou odesílány klientovi. Odběry mohou zůstat aktivní po dobu, kdy je klient odpojen. Publikování jsou odesílána klientovi při opětovném připojení.

Řetězce témat a filtry témat v klientech MQTT
Řetězce témat a filtry témat se používají k publikování a k odběru. Syntaxe řetězců témat a filtrů v klientech systému MQTT je z velké části stejná jako řetězce témat v produktu IBM MQ.

Vyčistit relace

Klient MQTT a služba telemetrie (MQXR) udržují informace o stavu relace. Informace o stavu se používají k zajištění "alespoň jednou" a "přesně jednou" doručení a "přesně jednou" přijetí publikování. 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 volby `MqttConnectOptions.cleanSession` před připojením.

Když připojujete klientskou aplikaci MQTT pomocí metody `MqttClient.connect` , klient identifikuje připojení pomocí 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 se předchozí informace o relaci na klientovi a serveru vymažou. Pokud je hodnota `cleanSession=false` , předchozí relace bude obnovena. Pokud neexistuje žádná předchozí relace, spustí se nová relace.

Poznámka: Administrátor produktu IBM MQ může vynuceně zavřít otevřenou relaci a odstranit všechny informace o relaci. Pokud klient znovu otevře relaci s produktem `cleanSession=false`, spustí se nová relace.

Publikace

Pokud před připojením klienta použijete výchozí hodnotu `MqttConnectOptions`nebo nastavíte hodnotu `MqttConnectOptions.cleanSession` na `true` , budou při připojení klienta odebrána všechna nevyřízená doručení publikování pro klienta.

Nastavení čisté relace nemá žádný vliv na publikování odesílaná s produktem `QoS=0`. Pro `QoS=1` a `QoS=2`může použití `cleanSession=true` vést ke ztrátě publikace.

Odběry

Pokud před připojením klienta použijete výchozí hodnotu `MqttConnectOptions`nebo nastavíte hodnotu `MqttConnectOptions.cleanSession` na `true` , budou při připojení klienta odebrány všechny staré odběry klienta. Všechny nové odběry, které klient během relace vytvoří, jsou při odpojení odebrány.

Pokud před připojením nastavíte hodnotu `MqttConnectOptions.cleanSession` na `false` , všechny odběry, které klient vytvoří, se přidají ke všem odběrům, které pro klienta existovaly před připojením. 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 po celou relaci. Chcete-li změnit jeho nastavení, je třeba klienta odpojit a znovu připojit. Změníte-li režimy z použití `cleanSession=false` na `cleanSession=true`, budou všechny předchozí odběry pro klienta a nepřijatá publikování zrušeny.

Související pojmy

Zpětná volání a synchronizace v klientských aplikacích MQTT
Programovací model klienta MQTT používá podprocesy ve velké míře. Podprocesy oddělují klientskou aplikaci systému MQTT v co největší míře od prodlev při přenosu zpráv na server a ze serveru. Publikování, tokeny doručení a události ztracených připojení jsou doručeny metodám 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 můžete použít libovolný identifikační řetězec. Je důležité mít proceduru pro přidělení identifikátorů klientů a prostředky pro konfiguraci klienta s jeho zvoleným identifikátorem.

Tokeny doručení

Poslední vůle a vyhlášení závěti
Pokud připojení klienta MQTT neočekávaně skončí, můžete nakonfigurovat produkt MQ Telemetry tak, aby odeslal publikaci "last will and testament". Předem definujte obsah publikace a téma, na které má být publikace odeslána. "Poslední vůle a závět" je vlastnost spojení. Vytvořte jej před připojením klienta.

Perzistence zpráv v klientech MQTT
Zprávy publikování jsou trvalé, pokud jsou odeslány s kvalitou služby "alespoň jednou"nebo "přesně jednou". Můžete implementovat vlastní mechanizmus perzistence na klientovi nebo použít výchozí mechanizmus perzistence, který je poskytován s klientem. Perzistence funguje v obou směrech pro publikace odeslané klientovi nebo z klienta.

Publikace
Publikace jsou instance produktu `MqttMessage` , které jsou přidruženy k řetězci tématu. Klienti systému MQTT mohou vytvářet publikování, která se odesílají do produktu IBM MQ, a odebírat témata týkající se produktu IBM MQ , která přijímají publikování.

Kvality služby poskytované klientem MQTT
Klient systému MQTT poskytuje tři kvality služeb pro doručování publikací produktu IBM MQ a klientu MQTT : "nejvýše jednou", "alespoň jednou" a "přesně jednou". Když klient MQTT odešle požadavek na vytvoření odběru do produktu IBM MQ , odešle se požadavek s kvalitou služby "alespoň jednou".

Zachovaná publikování a klienti MQTT
Téma může mít jedno a pouze jedno zachované publikování. Pokud vytvoříte odběr tématu, které má zachované publikování, publikování vám bude okamžitě předáno.

Odběry
Vytvořte odběry pro registraci zájmu o témata publikování pomocí filtru témat. Klient může vytvořit více odběrů nebo odběr obsahující filtr témat, který používá zástupné znaky k registraci zájmu o více témat. Publikace o tématech odpovídajících filtrům jsou odesílány klientovi. Odběry mohou zůstat aktivní po dobu, kdy je klient odpojen. Publikování jsou odesílána klientovi při opětovném připojení.

Řetězce témat a filtry témat v klientech MQTT
Řetězce témat a filtry témat se používají k publikování a k odběru. Syntaxe řetězců témat a filtrů v klientech systému MQTT je z velké části stejná 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 můžete použít libovolný identifikační řetězec. Je důležité mít proceduru pro přidělení identifikátorů klientů a prostředky pro konfiguraci klienta s jeho zvoleným identifikátorem.

Identifikátor klienta se používá v administraci systému MQTT . Vzhledem k tomu, že správu mohou provádět stovky tisíc klientů, musíte být schopni rychle identifikovat konkrétního klienta. Předpokládejme například, že zařízení nefunguje správně a vy jste upozorněni, například tím, že zákazník zazvoní na středisko podpory. Zákazník musí být schopen identifikovat zařízení a vy musíte být schopni korelovat tuto identifikaci se serverem, který je obvykle připojen ke klientovi.

Při procházení připojení klienta MQTT je každé připojení označeno identifikátorem klienta. Chcete-li se rozhodnout, jak nejlépe mapovat tento identifikátor na zařízení a server, zeptejte se sami sebe na následující otázky:

- Bylo by vhodné udržovat a používat databázi, která mapuje každé zařízení na identifikátor klienta a na server?
- Může název zařízení identifikovat server, ke kterému je připojeno?
- Potřebujete vyhledávací tabulku, která mapuje identifikátor klienta na fyzické zařízení?
- Identifikuje identifikátor klienta konkrétní zařízení, uživatele nebo aplikaci spuštěnou na klientovi?
- Pokud zákazník nahradí vadné zařízení novým, má nové zařízení stejný identifikátor jako staré zařízení, nebo přidělíte nový identifikátor? (Pokud změníte fyzické zařízení a zachováte stejný identifikátor, neprovedená publikování a aktivní odběry se automaticky přenesou na nové zařízení.)

Také potřebujete systém, abyste se ujistili, že jsou identifikátory klienta jedinečné, a musíte mít spolehlivý proces pro nastavení identifikátoru na klientovi. Pokud je klientské zařízení "black-box" bez uživatelského rozhraní, můžete zařízení vyrobit s identifikátorem klienta, nebo můžete mít proces instalace a konfigurace softwaru, který konfiguruje zařízení před jeho aktivací.

Chcete-li zachovat krátký a jedinečný identifikátor, můžete vytvořit identifikátor klienta z 48bitové adresy MAC zařízení. Pokud velikost přenosu není kritický problém, můžete použít zbývajících 17 bajtů, abyste usnadnili správu adresy.

Související pojmy

Zpětná volání a synchronizace v klientských aplikacích MQTT
Programovací model klienta MQTT používá podprocesy ve velké míře. Podprocesy oddělují klientskou aplikaci systému MQTT v co největší míře od prodlev při přenosu zpráv na server a ze serveru. Publikování, tokeny doručení a události ztracených připojení jsou doručeny metodám ve třídě zpětného volání, která implementuje `MqttCallback`.

Vyčistit relace
Klient MQTT a služba telemetrie (MQXR) udržují informace o stavu relace. Informace o stavu se používají k zajištění "alespoň jednou" a "přesně jednou" doručení a "přesně jednou" přijetí publikování. 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 volby `MqttConnectOptions.cleanSession` před připojením.

Tokeny doručení

Poslední vůle a vyhlášení závěti
Pokud připojení klienta MQTT neočekávaně skončí, můžete nakonfigurovat produkt MQ Telemetry tak, aby odeslal publikaci "last will and testament". Předem definujte obsah publikace a téma, na které má být publikace odeslána. "Poslední vůle a závěť" je vlastnost spojení. Vytvořte jej před připojením klienta.

Perzistence zpráv v klientech MQTT
Zprávy publikování jsou trvalé, pokud jsou odeslány s kvalitou služby "alespoň jednou"nebo "přesně jednou". Můžete implementovat vlastní mechanizmus perzistence na klientovi nebo použít výchozí mechanizmus perzistence, který je poskytován s klientem. Perzistence funguje v obou směrech pro publikace odeslané klientovi nebo z klienta.

Publikace
Publikace jsou instance produktu `MqttMessage` , které jsou přidruženy k řetězci tématu. Klienti systému MQTT mohou vytvářet publikování, která se odesílají do produktu IBM MQ, a odebírat témata týkající se produktu IBM MQ , která přijímají publikování.

Kvality služby poskytované klientem MQTT
Klient systému MQTT poskytuje tři kvality služeb pro doručování publikací produktu IBM MQ a klientu MQTT : "nejvýše jednou", "alespoň jednou" a "přesně jednou". Když klient MQTT odešle požadavek na vytvoření odběru do produktu IBM MQ , odešle se požadavek s kvalitou služby "alespoň jednou".

Zachovaná publikování a klienti MQTT
Téma může mít jedno a pouze jedno zachované publikování. Pokud vytvoříte odběr tématu, které má zachované publikování, publikování vám bude okamžitě předáno.

Odběry
Vytvořte odběry pro registraci zájmu o témata publikování pomocí filtru témat. Klient může vytvořit více odběrů nebo odběr obsahující filtr témat, který používá zástupné znaky k registraci zájmu o více témat. Publikace o tématech odpovídajících filtrům jsou odesílány klientovi. Odběry mohou zůstat aktivní po dobu, kdy je klient odpojen. Publikování jsou odesílána klientovi při opětovném připojení.

Řetězce témat a filtry témat v klientech MQTT
Řetězce témat a filtry témat se používají k publikování a k odběru. Syntaxe řetězců témat a filtrů v klientech systému MQTT je z velké části stejná jako řetězce témat v produktu IBM MQ.

Tokeny doručení

Když klient publikuje téma, vytvoří se nový token doručení. Token doručení použijte k monitorování doručení publikace nebo k blokování aplikace klienta, dokud nebude doručení dokončeno.

Token je objekt `MqttDeliveryToken` . Je vytvořen voláním metody MqttTopic.publish () a je zachován klientem MQTT , dokud není relace klienta odpojena a doručení dokončeno.

Obvyklým použitím tokenu je kontrola, zda je doručení dokončeno. Blokujte aplikaci klienta až do dokončení doručení pomocí vráceného tokenu pro volání `token.waitForCompletion`. Případně poskytněte obslužnou rutinu `MqttCallBack` . Když klient MQTT obdrží všechna potvrzení, která očekává jako součást doručení publikace, zavolá `MqttCallBack.deliveryComplete` a předá token doručení jako parametr.

Dokud nebude doručení dokončeno, můžete zkontrolovat publikování pomocí vráceného tokenu doručení voláním funkce `token.getMessage`.

Dokončené dodávky
Dokončení dodávek je asynchronní a závisí na kvalitě služby spojené s publikací.

Nanejvýš jednou

`QoS=0`
Doručení je dokončeno ihned po návratu z `MqttTopic.publish`. Funkce `MqttCallback.deliveryComplete` je volána okamžitě.

Alespoň jednou

`QoS=1`
Doručení je dokončeno po přijetí potvrzení pro publikování od správce front. Funkce `MqttCallback.deliveryComplete` je volána při přijetí potvrzení. Zpráva může být doručena více než jednou před zavoláním `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. Funkce `MqttCallback.deliveryComplete` je volána ihned po přijetí zprávy publikování. Nečeká na zprávu o dokončení.

Za výjimečných okolností se vaše klientská aplikace nemusí vrátit do klienta MQTT z produktu `MqttCallback.deliveryComplete` normálně. Víte, že doručení bylo dokončeno, protože byl volán `MqttCallback.deliveryComplete` . Pokud klient restartuje stejnou relaci, `MqttCallback.deliveryComplete` se znovu nevyvolá.

Neúplné dodávky

Pokud není doručení po odpojení relace klienta dokončeno, můžete klienta znovu připojit a dokončit doručení. Doručení 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 poté se znovu připojte a nastavte atribut `cleanSession` `MqttConnectionOptions` na hodnotu `false` . Pokud nastavíte volbu `cleanSession` na hodnotu `true`, budou nevyřízené tokeny doručení vyhozeny.

Můžete zkontrolovat, zda existují nevyřízené dodávky, voláním funkce `MqttClient.getPendingDeliveryTokens`. Před připojením klienta můžete volat `MqttClient.getPendingDeliveryTokens` .

Související pojmy

Zpětná volání a synchronizace v klientských aplikacích MQTT
Programovací model klienta MQTT používá podprocesy ve velké míře. Podprocesy oddělují klientskou aplikaci systému MQTT v co největší míře od prodlev při přenosu zpráv na server a ze serveru. Publikování, tokeny doručení a události ztracených připojení jsou doručeny metodám ve třídě zpětného volání, která implementuje `MqttCallback`.

Vyčistit relace
Klient MQTT a služba telemetrie (MQXR) udržují informace o stavu relace. Informace o stavu se používají k zajištění "alespoň jednou" a "přesně jednou" doručení a "přesně jednou" přijetí publikování. 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 volby `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 můžete použít libovolný identifikační řetězec. Je důležité mít proceduru pro přidělení identifikátorů klientů a prostředky pro konfiguraci klienta s jeho zvoleným identifikátorem.

Poslední vůle a vyhlášení závěti
Pokud připojení klienta MQTT neočekávaně skončí, můžete nakonfigurovat produkt MQ Telemetry tak, aby odeslal publikaci "last will and testament". Předem definujte obsah publikace a téma, na které má být publikace odeslána. "Poslední vůle a závěť" je vlastnost spojení. Vytvořte jej před připojením klienta.

Perzistence zpráv v klientech MQTT
Zprávy publikování jsou trvalé, pokud jsou odeslány s kvalitou služby "alespoň jednou"nebo "přesně jednou". Můžete implementovat vlastní mechanizmus perzistence na klientovi nebo použít výchozí mechanizmus perzistence, který je poskytován s klientem. Perzistence funguje v obou směrech pro publikace odeslané klientovi nebo z klienta.

Publikace
Publikace jsou instance produktu `MqttMessage` , které jsou přidruženy k řetězci tématu. Klienti systému MQTT mohou vytvářet publikování, která se odesílají do produktu IBM MQ, a odebírat témata týkající se produktu IBM MQ , která přijímají publikování.

Kvality služby poskytované klientem MQTT
Klient systému MQTT poskytuje tři kvality služeb pro doručování publikací produktu IBM MQ a klientu MQTT : "nejvýše jednou", "alespoň jednou" a "přesně jednou". Když klient MQTT odešle požadavek na vytvoření odběru do produktu IBM MQ , odešle se požadavek s kvalitou služby "alespoň jednou".

Zachovaná publikování a klienti MQTT
Téma může mít jedno a pouze jedno zachované publikování. Pokud vytvoříte odběr tématu, které má zachované publikování, publikování vám bude okamžitě předáno.

Odběry
Vytvořte odběry pro registraci zájmu o témata publikování pomocí filtru témat. Klient může vytvořit více odběrů nebo odběr obsahující filtr témat, který používá zástupné znaky k registraci zájmu o více témat. Publikace o tématech odpovídajících filtrům jsou odesílány klientovi. Odběry mohou zůstat aktivní po dobu, kdy je klient odpojen. Publikování jsou odesílána klientovi při opětovném připojení.

Řetězce témat a filtry témat v klientech MQTT
Řetězce témat a filtry témat se používají k publikování a k odběru. Syntaxe řetězců témat a filtrů v klientech systému MQTT je z velké části stejná jako řetězce témat v produktu IBM MQ.

Poslední vůle a vyhlášení závěti

Pokud připojení klienta MQTT neočekávaně skončí, můžete nakonfigurovat produkt MQ Telemetry tak, aby odeslal publikaci "last will and testament". Předem definujte obsah publikace a téma, na které má být publikace odeslána. "Poslední vůle a závěť" je vlastnost spojení. Vytvořte jej před připojením klienta.

Vytvořte téma pro poslední vůli a závěť. Můžete vytvořit téma, jako např. `MQTTManagement/Connections/`*`server URI`*`/`*`client identifer`*`/Lost`.

Nastavte "poslední vůle a závěť" pomocí metody `MqttConnectionOptions.setWill(MqttTopic lastWillTopic, byte [] lastWillPayload, int lastWillQos, boolean lastWillRetained)` .

Zvažte vytvoření časového razítka ve zprávě `lastWillPayload` . Zahrňte další informace o klientovi, které pomáhají identifikovat klienta a okolnosti připojení. Předejte objekt `MqttConnectionOptions` konstruktoru `MqttClient` .

Nastavte `lastWillQos` na hodnotu 1 nebo 2, chcete-li nastavit zprávu jako trvalou v produktu IBM MQa zaručit doručení. Chcete-li uchovat poslední ztracené informace o připojení, nastavte parametr `lastWillRetained` na hodnotu `true`.

Publikace "poslední vůle a závěť" je odeslána odběratelům, pokud připojení neočekávaně skončí. Odešle se, pokud připojení skončí bez klienta, který volá metodu `MqttClient.disconnect` .

Chcete-li monitorovat připojení, doplňte publikaci "poslední vůle a závěť" o další publikace pro zaznamenávání připojení a programovaná odpojení.

Související pojmy

Zpětná volání a synchronizace v klientských aplikacích MQTT
Programovací model klienta MQTT používá podprocesy ve velké míře. Podprocesy oddělují klientskou aplikaci systému MQTT v co největší míře od prodlev při přenosu zpráv na server a ze serveru. Publikování, tokeny doručení a události ztracených připojení jsou doručeny metodám ve třídě zpětného volání, která implementuje `MqttCallback`.

Vyčistit relace
Klient MQTT a služba telemetrie (MQXR) udržují informace o stavu relace. Informace o stavu se používají k zajištění "alespoň jednou" a "přesně jednou" doručení a "přesně jednou" přijetí publikování. 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 volby `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 můžete použít libovolný identifikační řetězec. Je důležité mít proceduru pro přidělení identifikátorů klientů a prostředky pro konfiguraci klienta s jeho zvoleným identifikátorem.

Tokeny doručení

Perzistence zpráv v klientech MQTT
Zprávy publikování jsou trvalé, pokud jsou odeslány s kvalitou služby "alespoň jednou"nebo "přesně jednou". Můžete implementovat vlastní mechanizmus perzistence na klientovi nebo použít výchozí mechanizmus perzistence, který je poskytován s klientem. Perzistence funguje v obou směrech pro publikace odeslané klientovi nebo z klienta.

Publikace
Publikace jsou instance produktu `MqttMessage` , které jsou přidruženy k řetězci tématu. Klienti systému MQTT mohou vytvářet publikování, která se odesílají do produktu IBM MQ, a odebírat témata týkající se produktu IBM MQ , která přijímají publikování.

Kvality služby poskytované klientem MQTT
Klient systému MQTT poskytuje tři kvality služeb pro doručování publikací produktu IBM MQ a klientu MQTT : "nejvýše jednou", "alespoň jednou" a "přesně jednou". Když klient MQTT odešle požadavek na vytvoření odběru do produktu IBM MQ , odešle se požadavek s kvalitou služby "alespoň jednou".

Zachovaná publikování a klienti MQTT
Téma může mít jedno a pouze jedno zachované publikování. Pokud vytvoříte odběr tématu, které má zachované publikování, publikování vám bude okamžitě předáno.

Odběry
Vytvořte odběry pro registraci zájmu o témata publikování pomocí filtru témat. Klient může vytvořit více odběrů nebo odběr obsahující filtr témat, který používá zástupné znaky k registraci zájmu o více témat. Publikace o tématech odpovídajících filtrům jsou odesílány klientovi. Odběry mohou zůstat aktivní po dobu, kdy je klient odpojen. Publikování jsou odesílána klientovi při opětovném připojení.

Řetězce témat a filtry témat v klientech MQTT
Řetězce témat a filtry témat se používají k publikování a k odběru. Syntaxe řetězců témat a filtrů v klientech systému MQTT je z velké části stejná jako řetězce témat v produktu IBM MQ.

Perzistence zpráv v klientech MQTT

Zprávy publikování jsou trvalé, pokud jsou odeslány s kvalitou služby "alespoň jednou"nebo "přesně jednou". Můžete implementovat vlastní mechanizmus perzistence na klientovi nebo použít výchozí mechanizmus perzistence, který je poskytován s klientem. Perzistence funguje v obou směrech pro publikace odeslané klientovi nebo z klienta.

V produktu MQTTmá perzistence zpráv dva aspekty: jak je zpráva přenášena a zda je zařazena do fronty v produktu IBM MQ jako trvalá zpráva.

1. Klient MQTT spojí perzistenci zpráv s kvalitou služby. V závislosti na kvalitě služby, kterou jste zvolili pro zprávu, je zpráva trvalá. Perzistence zpráv je nezbytná pro implementaci požadované kvality služby.

   Zadáte-li "nejvýše jednou", QoS=0, klient vyřadí zprávu ihned po jejím publikování. Pokud dojde k selhání v předchozím zpracování zprávy, zpráva se znovu neodešle. I když klient zůstane aktivní, zpráva se znovu neodešle. Chování zpráv QoS=0 je stejné jako u rychlých přechodných zpráv IBM MQ .

   Pokud je zpráva publikována klientem s QoS 1 nebo 2, je trvalá. Zpráva je uložena lokálně a je vyřazena z klienta pouze v případě, že již není potřeba zaručit "alespoň jednou", QoS=1nebo "přesně jednou", QoS=2doručení.
2. Pokud je zpráva označena jako QoS 1 nebo 2, je zařazena do fronty IBM MQ jako trvalá zpráva. Je-li označena jako QoS=0, bude zařazena do fronty v produktu IBM MQ jako dočasná zpráva. V produktu IBM MQ jsou přechodné zprávy přenášeny mezi správci front "přesně jednou", pokud kanál zpráv nemá nastaven atribut NPMSPEED na hodnotu FAST.

Trvalé publikování je uloženo v klientu, dokud není přijato klientskou aplikací. V případě systému QoS=2je publikování vyřazeno z klienta, když zpětné volání aplikace vrátí ovládací prvek. V případě systému QoS=1 může aplikace obdržet publikování znovu, dojde-li k selhání. V případě systému QoS=0zpětné volání neobdrží publikování více než jednou. Publikování nemusí být doručeno, pokud dojde k selhání, nebo pokud je klient v době publikování odpojen.

Když se přihlásíte k odběru tématu, můžete snížit hodnotu QoS , se kterou odběratel přijímá zprávy, aby odpovídal jeho schopnostem perzistence. Publikování vytvořená s vyšší hodnotou QoS jsou odesílána s nejvyšší hodnotou QoS , kterou požadoval odběratel.

Ukládání zpráv

Implementace ukládání dat na malých zařízeních se velmi liší. Model dočasného uložení trvalých zpráv v úložišti, které je spravováno klientem MQTT , může být příliš pomalý nebo vyžadovat příliš mnoho úložiště. Na mobilních zařízeních může mobilní operační systém poskytovat službu úložiště, která je ideální pro zprávy systému MQTT .

Pro zajištění flexibility při plnění omezení malých zařízení má klient MQTT dvě rozhraní perzistence. Rozhraní definují operace, které jsou zapojeny do ukládání trvalých zpráv. Rozhraní jsou popsána v dokumentaci k rozhraní API pro MQTT client for Java. Odkazy na dokumentaci rozhraní API klienta pro knihovny klienta MQTT viz Referenční příručka programování klienta MQTT. Rozhraní můžete

implementovat tak, aby vyhovovala zařízení. Klient MQTT , který je spuštěn na produktu Java SE, má výchozí implementaci rozhraní, která ukládají trvalé zprávy v systému souborů. Používá balík `java.io` .

Třídy perzistence

`MqttClientPersistence`
Předejte instanci implementace `MqttClientPersistence` klientovi MQTT jako parametr konstruktoru `MqttClient` . Pokud vynecháte parametr `MqttClientPersistence` z konstruktoru `MqttClient` , klient MQTT uloží trvalé zprávy pomocí třídy `MqttDefaultFilePersistence`.

`MqttPersistable`
`MqttClientPersistence` získá a vloží `MqttPersistable` objekty pomocí klíče úložiště. Pokud nepoužíváte produkt `MqttDefaultFilePersistence`, musíte poskytnout implementaci produktu `MqttPersistable` i implementaci produktu `MqttClientPersistence` .

`MqttDefaultFilePersistence`

Klient MQTT poskytuje třídu `MqttDefaultFilePersistence` . Pokud ve své klientské aplikaci převedete na instanci `MqttDefaultFilePersistence` , můžete zadat adresář pro ukládání trvalých zpráv jako parametr konstruktoru `MqttDefaultFilePersistence` .

Alternativně může klient MQTT vytvořit instanci `MqttDefaultFilePersistence` a umístit soubory do následujícího výchozího adresáře:

`client identifier` `-tcp` *`hostname portnumber`*

Následující znaky jsou odebrány z řetězce názvu adresáře:

`"\", "\\", "/", ":" a " "`

Cesta k adresáři je hodnota systémové vlastnosti `rcp.data`; Není-li `rcp.data` nastavena, cesta je hodnotou systémové vlastnosti `usr.data`, kde

- `rcp.data` je vlastnost přidružená k instalaci platformy RCP (OSGi nebo Eclipse Rich Client Platform).
- `usr.data` je adresář, ve kterém byl spuštěn příkaz Java , který spustil aplikaci.

Související pojmy
Zpětná volání a synchronizace v klientských aplikacích MQTT
Programovací model klienta MQTT používá podprocesy ve velké míře. Podprocesy oddělují klientskou aplikaci systému MQTT v co největší míře od prodlev při přenosu zpráv na server a ze serveru. Publikování, tokeny doručení a události ztracených připojení jsou doručeny metodám ve třídě zpětného volání, která implementuje `MqttCallback`.

Vyčistit relace
Klient MQTT a služba telemetrie (MQXR) udržují informace o stavu relace. Informace o stavu se používají k zajištění "alespoň jednou" a "přesně jednou" doručení a "přesně jednou" přijetí publikování. 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 volby `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 můžete použít libovolný identifikační řetězec. Je důležité mít proceduru pro přidělení identifikátorů klientů a prostředky pro konfiguraci klienta s jeho zvoleným identifikátorem.

Tokeny doručení

Poslední vůle a vyhlášení závěti
Pokud připojení klienta MQTT neočekávaně skončí, můžete nakonfigurovat produkt MQ Telemetry tak, aby odeslal publikaci "last will and testament". Předem definujte obsah publikace a téma, na které má být publikace odeslána. "Poslední vůle a závěť" je vlastnost spojení. Vytvořte jej před připojením klienta.

Publikace
Publikace jsou instance produktu MqttMessage , které jsou přidruženy k řetězci tématu. Klienti systému MQTT mohou vytvářet publikování, která se odesílají do produktu IBM MQ, a odebírat témata týkající se produktu IBM MQ , která přijímají publikování.

Kvality služby poskytované klientem MQTT
Klient systému MQTT poskytuje tři kvality služeb pro doručování publikací produktu IBM MQ a klientu MQTT : "nejvýše jednou", "alespoň jednou" a "přesně jednou". Když klient MQTT odešle požadavek na vytvoření odběru do produktu IBM MQ , odešle se požadavek s kvalitou služby "alespoň jednou".

Zachovaná publikování a klienti MQTT
Téma může mít jedno a pouze jedno zachované publikování. Pokud vytvoříte odběr tématu, které má zachované publikování, publikování vám bude okamžitě předáno.

Odběry
Vytvořte odběry pro registraci zájmu o témata publikování pomocí filtru témat. Klient může vytvořit více odběrů nebo odběr obsahující filtr témat, který používá zástupné znaky k registraci zájmu o více témat. Publikace o tématech odpovídajících filtrům jsou odesílány klientovi. Odběry mohou zůstat aktivní po dobu, kdy je klient odpojen. Publikování jsou odesílána klientovi při opětovném připojení.

Řetězce témat a filtry témat v klientech MQTT
Řetězce témat a filtry témat se používají k publikování a k odběru. Syntaxe řetězců témat a filtrů v klientech systému MQTT je z velké části stejná jako řetězce témat v produktu IBM MQ.

Publikace

Publikace jsou instance produktu MqttMessage , které jsou přidruženy k řetězci tématu. Klienti systému MQTT mohou vytvářet publikování, která se odesílají do produktu IBM MQ, a odebírat témata týkající se produktu IBM MQ , která přijímají publikování.

MqttMessage má jako svůj informační obsah bajtové pole. Cílem je udržet zprávy tak malé, jak je to možné. Maximální délka zprávy povolená MQTT protocol je 250 MB.

Klientský program systému MQTT obvykle používá k manipulaci s obsahem zprávy java.lang.String nebo java.lang.StringBuffer . Pro usnadnění má třída MqttMessage metodu toString pro převod svého informačního obsahu na řetězec. Chcete-li vytvořit informační obsah bajtového pole z java.lang.String nebo java.lang.StringBuffer, použijte metodu getBytes .

Metoda getBytes převede řetězec na výchozí znakovou sadu pro platformu. Výchozí znaková sada je obecně UTF-8. Publikace MQTT , které obsahují pouze text, jsou obvykle zakódovány v souboru UTF-8. Pomocí metody getBytes("UTF8") potlačte výchozí znakovou sadu.

V produktu IBM MQje publikace MQTT přijata jako zpráva jms-bytes . Zpráva obsahuje složku MQRFH2 obsahující složku <mqtt>a složku <mqps> . Složka <mqtt> obsahuje clientId, msgIda qos, ale tento obsah se může v budoucnu změnit.

MqttMessage má tři další atributy: kvalitu služby, zda je zachována a zda je duplicitní. Duplicitní příznak je nastaven pouze v případě, že kvalita služby je "alespoň jednou" nebo "přesně jednou". Pokud byla zpráva odeslána dříve a nebyla dostatečně rychle potvrzena klientem MQTT , zpráva se odešle znovu s duplicitním atributem nastaveným na hodnotu true.

Publikování

Chcete-li vytvořit publikování v aplikaci klienta MQTT , vytvořte MqttMessage. Nastavte jeho informační obsah, kvalitu služby a to, zda je zachován, a zavolejte metodu MqttTopic.publish(MqttMessage message) ; MqttDeliveryToken je vrácena a dokončení publikování je asynchronní.

Klient MQTT může také vytvořit dočasný objekt zprávy z parametrů metody MqttTopic.publish(byte [] payload, int qos, boolean retained) při vytváření publikování.

Pokud má publikace "alespoň jednou" nebo "právě jednou" kvalitu služby QoS=1 nebo QoS=2, klient MQTT volá rozhraní MqttClientPersistence . Volá MqttClientPersistence k uložení zprávy před vrácením tokenu doručení 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` ihned po doručení publikování. V závislosti na kvalitě služby může být doručení pro produkt `QoS=0`téměř okamžité, nebo může trvat určitou dobu pro produkt `QoS=2`.

Použijte metodu `MqttDeliveryToken.isComplete` k vyzvání, pokud je doručení dokončeno. Zatímco hodnota `MqttDeliveryToken.isComplete` je `false`, můžete volat `MqttDeliveryToken.getMessage` , abyste získali obsah zprávy. Pokud je výsledkem volání `MqttDeliveryToken.isComplete` `true`, zpráva byla zrušena a volání `MqttDeliveryToken.getMessage` by vyvolalo výjimku ukazatele Null. Mezi `MqttDeliveryToken.getMessage` a `MqttDeliveryToken.isComplete`neexistuje žádná vestavěná synchronizace.

Pokud se klient odpojí před přijetím všech nevyřízených tokenů doručení, může se nová instance klienta před připojením dotazovat na nevyřízené tokeny doručení. Dokud se klient nepřipojí, nejsou dokončena žádná nová doručení a je bezpečné volat `MqttDeliveryToken.getMessage`. Pomocí metody `MqttDeliveryToken.getMessage` zjistěte, která publikování nebyla dodána. Nevyřízené tokeny doručení jsou vyřazeny, pokud se připojujete s hodnotou `MqttConnectOptions.cleanSession` nastavenou na výchozí hodnotu `true`.

přihlášení odběru

Správce front je odpovědný za vytváření publikování, která mají být odeslána odběrateli 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 obsahovat zástupné znaky. Před předáním publikování odběrateli správcem front zkontroluje správce front atributy tématu přidružené k publikování. Podle postupu vyhledávání popsaného v tématu Přihlášení k odběru pomocí řetězce tématu, který obsahuje zástupné znaky , identifikuje, zda objekt administrativního tématu uděluje uživateli oprávnění k odběru.

Když klient MQTT obdrží publikaci s kvalitou služby "alespoň jednou", volá metodu `MqttCallback.messageArrived` ke zpracování publikace. Pokud je kvalita služby publikování "právě jednou", `QoS=2`, klient MQTT volá rozhraní `MqttClientPersistence` , aby uložil zprávu při jejím přijetí. Pak volá `MqttCallback.messageArrived`.

Související pojmy

Zpětná volání a synchronizace v klientských aplikacích MQTT
Programovací model klienta MQTT používá podprocesy ve velké míře. Podprocesy oddělují klientskou aplikaci systému MQTT v co největší míře od prodlev při přenosu zpráv na server a ze serveru. Publikování, tokeny doručení a události ztracených připojení jsou doručeny metodám ve třídě zpětného volání, která implementuje `MqttCallback`.

Vyčistit relace
Klient MQTT a služba telemetrie (MQXR) udržují informace o stavu relace. Informace o stavu se používají k zajištění "alespoň jednou" a "přesně jednou" doručení a "přesně jednou" přijetí publikování. 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 volby `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 můžete použít libovolný identifikační řetězec. Je důležité mít proceduru pro přidělení identifikátorů klientů a prostředky pro konfiguraci klienta s jeho zvoleným identifikátorem.

Tokeny doručení

Poslední vůle a vyhlášení závěti

Pokud připojení klienta MQTT neočekávaně skončí, můžete nakonfigurovat produkt MQ Telemetry tak, aby odeslal publikaci "last will and testament". Předem definujte obsah publikace a téma, na které má být publikace odeslána. "Poslední vůle a závěť" je vlastnost spojení. Vytvořte jej před připojením klienta.

Perzistence zpráv v klientech MQTT
Zprávy publikování jsou trvalé, pokud jsou odeslány s kvalitou služby "alespoň jednou"nebo "přesně jednou". Můžete implementovat vlastní mechanizmus perzistence na klientovi nebo použít výchozí mechanizmus perzistence, který je poskytován s klientem. Perzistence funguje v obou směrech pro publikace odeslané klientovi nebo z klienta.

Kvality služby poskytované klientem MQTT
Klient systému MQTT poskytuje tři kvality služeb pro doručování publikací produktu IBM MQ a klientu MQTT : "nejvýše jednou", "alespoň jednou" a "přesně jednou". Když klient MQTT odešle požadavek na vytvoření odběru do produktu IBM MQ , odešle se požadavek s kvalitou služby "alespoň jednou".

Zachovaná publikování a klienti MQTT
Téma může mít jedno a pouze jedno zachované publikování. Pokud vytvoříte odběr tématu, které má zachované publikování, publikování vám bude okamžitě předáno.

Odběry
Vytvořte odběry pro registraci zájmu o témata publikování pomocí filtru témat. Klient může vytvořit více odběrů nebo odběr obsahující filtr témat, který používá zástupné znaky k registraci zájmu o více témat. Publikace o tématech odpovídajících filtrům jsou odesílány klientovi. Odběry mohou zůstat aktivní po dobu, kdy je klient odpojen. Publikování jsou odesílána klientovi při opětovném připojení.

Řetězce témat a filtry témat v klientech MQTT
Řetězce témat a filtry témat se používají k publikování a k odběru. Syntaxe řetězců témat a filtrů v klientech systému MQTT je z velké části stejná jako řetězce témat v produktu IBM MQ.

Kvality služby poskytované klientem MQTT

Klient systému MQTT poskytuje tři kvality služeb pro doručování publikací produktu IBM MQ a klientu MQTT : "nejvýše jednou", "alespoň jednou" a "přesně jednou". Když klient MQTT odešle požadavek na vytvoření odběru do produktu IBM MQ , odešle se požadavek s kvalitou služby "alespoň jednou".

Kvalita služby publikace je atributem `MqttMessage`. Nastavuje se metodou `MqttMessage.setQos`.

Metoda `MqttClient.subscribe` může snížit kvalitu služby použité na publikování odesílaná klientovi v rámci tématu. Kvalita služby publikace předané odběrateli se může lišit od kvality služby publikace. Dolní z těchto dvou hodnot se používá k postoupení publikace.

Nanejvýš 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. To je někdy nazýváno "oheň a zapomenout".
Produkt MQTT protocol nevyžaduje, aby servery předávaly publikování v produktu `QoS=0` klientovi. Pokud je klient odpojen v době, kdy server obdrží publikování, může být publikování v závislosti na serveru zrušeno. Služba telemetrie (MQXR) nezrušit zprávy odeslané s produktem `QoS=0`. Jsou uloženy jako přechodné zprávy a jsou vyřazeny pouze v případě, že se správce front zastaví.

Nejméně jednou
`QoS=1`

`QoS=1` je výchozí režim přenosu.
Zpráva je vždy doručena alespoň jednou. Pokud odesilatel neobdrží potvrzení, bude zpráva znovu odeslána s nastaveným příznakem `DUP` , dokud nebude přijato potvrzení. V důsledku toho může být příjemce odeslán stejnou zprávu vícekrát a může ji zpracovat vícekrát.
Zpráva musí být uložena lokálně u odesílatele a příjemce, dokud není zpracována.
Zpráva je odstraněna z příjemce poté, co byla zpracována. Pokud je příjemcem zprostředkovatel, zpráva se publikuje svým odběratelům. Pokud je příjemcem klient, je zpráva doručena aplikaci odběratele. Po odstranění zprávy příjemce odešle potvrzení odesílateli.

Zpráva je odstraněna z odesílatele poté, co obdrží potvrzení od příjemce.

Přesně jednou
QoS=2

Zpráva je vždy doručena přesně jednou.

Zpráva musí být uložena lokálně u odesílatele a příjemce, dokud není zpracována.

QoS=2 je nejbezpečnější, ale nejpomalejší režim přenosu. Před odstraněním zprávy od odesilatele trvá mezi odesilatelem a příjemcem alespoň dvě dvojice přenosů. Zprávu lze zpracovat v přijímači po prvním přenosu.

V první dvojici přenosů odesilatel odešle zprávu a obdrží od příjemce potvrzení, že zprávu uložil. Pokud odesilatel neobdrží potvrzení, bude zpráva znovu odeslána s nastaveným příznakem DUP , dokud nebude přijato potvrzení.

Ve druhé dvojici přenosů sdělí odesilatel příjemci, že může dokončit zpracování zprávy "PUBREL". Pokud odesilatel neobdrží potvrzení zprávy "PUBREL" , bude zpráva "PUBREL" odeslána znovu, dokud nebude přijato potvrzení. Odesílatel odstraní zprávu, kterou uložil, když obdrží potvrzení pro zprávu "PUBREL" .

Příjemce může zpracovat zprávu v první nebo druhé fázi za předpokladu, že zprávu znovu nezpracuje. Pokud je příjemcem zprostředkovatel, publikuje zprávu odběratelům. Pokud je příjemcem klient, doručí zprávu aplikaci odběratele. Příjemce odešle zprávu o dokončení zpět odesílateli, že dokončil zpracování zprávy.

Související pojmy

Zpětná volání a synchronizace v klientských aplikacích MQTT
Programovací model klienta MQTT používá podprocesy ve velké míře. Podprocesy oddělují klientskou aplikaci systému MQTT v co největší míře od prodlev při přenosu zpráv na server a ze serveru. Publikování, tokeny doručení a události ztracených připojení jsou doručeny metodám ve třídě zpětného volání, která implementuje `MqttCallback`.

Vyčistit relace
Klient MQTT a služba telemetrie (MQXR) udržují informace o stavu relace. Informace o stavu se používají k zajištění "alespoň jednou" a "přesně jednou" doručení a "přesně jednou" přijetí publikování. 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 volby `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 můžete použít libovolný identifikační řetězec. Je důležité mít proceduru pro přidělení identifikátorů klientů a prostředky pro konfiguraci klienta s jeho zvoleným identifikátorem.

Tokeny doručení

Poslední vůle a vyhlášení závěti
Pokud připojení klienta MQTT neočekávaně skončí, můžete nakonfigurovat produkt MQ Telemetry tak, aby odeslal publikaci "last will and testament". Předem definujte obsah publikace a téma, na které má být publikace odeslána. "Poslední vůle a závět" je vlastnost spojení. Vytvořte jej před připojením klienta.

Perzistence zpráv v klientech MQTT
Zprávy publikování jsou trvalé, pokud jsou odeslány s kvalitou služby "alespoň jednou"nebo "přesně jednou". Můžete implementovat vlastní mechanizmus perzistence na klientovi nebo použít výchozí mechanizmus perzistence, který je poskytován s klientem. Perzistence funguje v obou směrech pro publikace odeslané klientovi nebo z klienta.

Publikace
Publikace jsou instance produktu `MqttMessage` , které jsou přidruženy k řetězci tématu. Klienti systému MQTT mohou vytvářet publikování, která se odesílají do produktu IBM MQ, a odebírat témata týkající se produktu IBM MQ , která přijímají publikování.

Zachovaná publikování a klienti MQTT

Téma může mít jedno a pouze jedno zachované publikování. Pokud vytvoříte odběr tématu, které má zachované publikování, publikování vám bude okamžitě předáno.

Odběry
Vytvořte odběry pro registraci zájmu o témata publikování pomocí filtru témat. Klient může vytvořit více odběrů nebo odběr obsahující filtr témat, který používá zástupné znaky k registraci zájmu o více témat. Publikace o tématech odpovídajících filtrům jsou odesílány klientovi. Odběry mohou zůstat aktivní po dobu, kdy je klient odpojen. Publikování jsou odesílána klientovi při opětovném připojení.

Řetězce témat a filtry témat v klientech MQTT
Řetězce témat a filtry témat se používají k publikování a k odběru. Syntaxe řetězců témat a filtrů v klientech systému MQTT je z velké části stejná jako řetězce témat v produktu IBM MQ.

Zachovaná publikování a klienti MQTT

Téma může mít jedno a pouze jedno zachované publikování. Pokud vytvoříte odběr tématu, které má zachované publikování, publikování vám bude okamžitě předáno.

Pomocí metody `MqttMessage.setRetained` určete, zda má být zachováno publikování tématu.

Když vytvoříte nebo aktualizujete zachované publikování, odešlete je s QoS 1 nebo 2. Pokud jej odešlete s hodnotou QoS 0, produkt IBM MQ vytvoří přechodné zachované publikování. Publikování nebude zachováno, pokud se správce front zastaví.

Pokud publikujete nezachované publikování do tématu, které má zachované publikování, toto zachované publikování nebude ovlivněno. Aktuální odběratelé obdrží nové publikování. Noví odběratelé obdrží nejprve zachované publikování a poté všechna nová publikování.

Chcete-li zaznamenat nejnovější hodnotu měření, můžete použít zachované publikování. Noví odběratelé tématu okamžitě obdrží nejnovější hodnotu měření. Pokud se od posledního přihlášení odběratele k odběru tématu publikování neprovedou žádná nová měření a odběratel se znovu přihlásí k odběru, obdrží odběratel znovu nejnovější zachované publikování pro dané téma.

Chcete-li odstranit zachované publikování, máte dvě možnosti:

- Spusťte příkaz **CLEAR TOPICSTR** MQSC.
- Vytvořte zachované publikování s nulovou délkou. Jak je uvedeno ve specifikaci MQTT 3.1.1 , pokud je zachovaná zpráva s nulovou délkou publikována v tématu, je jakákoli uchovaná zpráva pro toto téma vymazána.

Související pojmy
Zpětná volání a synchronizace v klientských aplikacích MQTT
Programovací model klienta MQTT používá podprocesy ve velké míře. Podprocesy oddělují klientskou aplikaci systému MQTT v co největší míře od prodlev při přenosu zpráv na server a ze serveru. Publikování, tokeny doručení a události ztracených připojení jsou doručeny metodám ve třídě zpětného volání, která implementuje `MqttCallback`.

Vyčistit relace
Klient MQTT a služba telemetrie (MQXR) udržují informace o stavu relace. Informace o stavu se používají k zajištění "alespoň jednou" a "přesně jednou" doručení a "přesně jednou" přijetí publikování. 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 volby `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 můžete použít libovolný identifikační řetězec. Je důležité mít proceduru pro přidělení identifikátorů klientů a prostředky pro konfiguraci klienta s jeho zvoleným identifikátorem.

Tokeny doručení
Poslední vůle a vyhlášení závěti

Pokud připojení klienta MQTT neočekávaně skončí, můžete nakonfigurovat produkt MQ Telemetry tak, aby odeslal publikaci "last will and testament". Předem definujte obsah publikace a téma, na které má být publikace odeslána. "Poslední vůle a závěť" je vlastnost spojení. Vytvořte jej před připojením klienta.

Perzistence zpráv v klientech MQTT
Zprávy publikování jsou trvalé, pokud jsou odeslány s kvalitou služby "alespoň jednou"nebo "přesně jednou". Můžete implementovat vlastní mechanizmus perzistence na klientovi nebo použít výchozí mechanizmus perzistence, který je poskytován s klientem. Perzistence funguje v obou směrech pro publikace odeslané klientovi nebo z klienta.

Publikace
Publikace jsou instance produktu `MqttMessage` , které jsou přidruženy k řetězci tématu. Klienti systému MQTT mohou vytvářet publikování, která se odesílají do produktu IBM MQ, a odebírat témata týkající se produktu IBM MQ , která přijímají publikování.

Kvality služby poskytované klientem MQTT
Klient systému MQTT poskytuje tři kvality služeb pro doručování publikací produktu IBM MQ a klientu MQTT : "nejvýše jednou", "alespoň jednou" a "přesně jednou". Když klient MQTT odešle požadavek na vytvoření odběru do produktu IBM MQ , odešle se požadavek s kvalitou služby "alespoň jednou".

Odběry
Vytvořte odběry pro registraci zájmu o témata publikování pomocí filtru témat. Klient může vytvořit více odběrů nebo odběr obsahující filtr témat, který používá zástupné znaky k registraci zájmu o více témat. Publikace o tématech odpovídajících filtrům jsou odesílány klientovi. Odběry mohou zůstat aktivní po dobu, kdy je klient odpojen. Publikování jsou odesílána klientovi při opětovném připojení.

Řetězce témat a filtry témat v klientech MQTT
Řetězce témat a filtry témat se používají k publikování a k odběru. Syntaxe řetězců témat a filtrů v klientech systému MQTT je z velké části stejná jako řetězce témat v produktu IBM MQ.

Odběry

Vytvořte odběry pro registraci zájmu o témata publikování pomocí filtru témat. Klient může vytvořit více odběrů nebo odběr obsahující filtr témat, který používá zástupné znaky k registraci zájmu o více témat. Publikace o tématech odpovídajících filtrům jsou odesílány klientovi. Odběry mohou zůstat aktivní po dobu, kdy je klient odpojen. Publikování jsou odesílána klientovi při opětovném připojení.

Vytvořte odběry pomocí metod `MqttClient.subscribe` , které předávají jeden nebo více filtrů témat a parametry kvality služby. Parametr kvality služby nastavuje maximální kvalitu služby, kterou je odběratel připraven použít k přijetí zprávy. Zprávy odeslané tomuto klientovi nelze doručit s vyšší kvalitou služby. Kvalita služby je nastavena na nižší z původních hodnot při publikování zprávy a na úroveň určenou pro odběr. Výchozí kvalita služby pro příjem zpráv je `QoS=1`, alespoň jednou.

Samotný požadavek na odběr je odeslán s `QoS=1`.

Publikování jsou přijímána odběratelem, když klient MQTT volá metodu `MqttCallback.messageArrived` . Metoda `messageArrived` také předá řetězec tématu, se kterým byla zpráva publikována, odběrateli.

Odběr nebo sadu či odběry můžete odebrat pomocí metod `MqttClient.unsubscribe` .

Příkaz IBM MQ může odebrat odběr. Vypište odběry pomocí produktu IBM MQ Explorernebo pomocí příkazů **runmqsc** či PCF. Všechny odběry klienta MQTT mají název. Dostanou název formuláře: *`ClientIdentifier`*:*`Topic name`*

Pokud před připojením klienta použijete výchozí hodnotu `MqttConnectOptions`nebo nastavíte hodnotu `MqttConnectOptions.cleanSession` na `true` , budou při připojení klienta odebrány všechny staré odběry klienta. Všechny nové odběry, které klient během relace vytvoří, jsou při odpojení odebrány.

Pokud před připojením nastavíte hodnotu `MqttConnectOptions.cleanSession` na `false` , všechny odběry, které klient vytvoří, se přidají ke všem odběrům, které pro klienta existovaly před připojením. 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 po celou relaci. Chcete-li změnit jeho nastavení, je třeba klienta odpojit a znovu připojit. Změníte-li režimy z použití `cleanSession=false` na `cleanSession=true`, budou všechny předchozí odběry pro klienta a nepřijatá publikování zrušeny.

Publikace, které se shodují s aktivními odběry, jsou odesílány klientovi ihned po jejich publikování. Pokud je klient odpojen, odešle se klientovi, pokud se znovu připojí ke stejnému serveru se stejným identifikátorem klienta a `MqttConnectOptions.cleanSession` nastaví na hodnotu `false`.

Odběry pro konkrétního klienta jsou identifikovány identifikátorem klienta. Můžete znovu připojit klienta z jiného klientského zařízení ke stejnému serveru a pokračovat se stejnými odběry a přijímat nedoručená publikování.

Související pojmy

Zpětná volání a synchronizace v klientských aplikacích MQTT
Programovací model klienta MQTT používá podprocesy ve velké míře. Podprocesy oddělují klientskou aplikaci systému MQTT v co největší míře od prodlev při přenosu zpráv na server a ze serveru. Publikování, tokeny doručení a události ztracených připojení jsou doručeny metodám ve třídě zpětného volání, která implementuje `MqttCallback`.

Vyčistit relace
Klient MQTT a služba telemetrie (MQXR) udržují informace o stavu relace. Informace o stavu se používají k zajištění "alespoň jednou" a "přesně jednou" doručení a "přesně jednou" přijetí publikování. 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 volby `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 můžete použít libovolný identifikační řetězec. Je důležité mít proceduru pro přidělení identifikátorů klientů a prostředky pro konfiguraci klienta s jeho zvoleným identifikátorem.

Tokeny doručení

Poslední vůle a vyhlášení závěti
Pokud připojení klienta MQTT neočekávaně skončí, můžete nakonfigurovat produkt MQ Telemetry tak, aby odeslal publikaci "last will and testament". Předem definujte obsah publikace a téma, na které má být publikace odeslána. "Poslední vůle a závěť" je vlastnost spojení. Vytvořte jej před připojením klienta.

Perzistence zpráv v klientech MQTT
Zprávy publikování jsou trvalé, pokud jsou odeslány s kvalitou služby "alespoň jednou"nebo "přesně jednou". Můžete implementovat vlastní mechanizmus perzistence na klientovi nebo použít výchozí mechanizmus perzistence, který je poskytován s klientem. Perzistence funguje v obou směrech pro publikace odeslané klientovi nebo z klienta.

Publikace
Publikace jsou instance produktu `MqttMessage` , které jsou přidruženy k řetězci tématu. Klienti systému MQTT mohou vytvářet publikování, která se odesílají do produktu IBM MQ, a odebírat témata týkající se produktu IBM MQ , která přijímají publikování.

Kvality služby poskytované klientem MQTT
Klient systému MQTT poskytuje tři kvality služeb pro doručování publikací produktu IBM MQ a klientu MQTT : "nejvýše jednou", "alespoň jednou" a "přesně jednou". Když klient MQTT odešle požadavek na vytvoření odběru do produktu IBM MQ , odešle se požadavek s kvalitou služby "alespoň jednou".

Zachovaná publikování a klienti MQTT
Téma může mít jedno a pouze jedno zachované publikování. Pokud vytvoříte odběr tématu, které má zachované publikování, publikování vám bude okamžitě předáno.

Řetězce témat a filtry témat v klientech MQTT

Řetězce témat a filtry témat se používají k publikování a k odběru. Syntaxe řetězců témat a filtrů v klientech systému MQTT je z velké části stejná jako řetězce témat v produktu IBM MQ.

Řetězce témat a filtry témat v klientech MQTT

Řetězce témat a filtry témat se používají k publikování a k odběru. Syntaxe řetězců témat a filtrů v klientech systému MQTT je z velké části stejná jako řetězce témat v produktu IBM MQ.

Řetězce témat se používají k odesílání publikací odběratelům. Vytvořte řetězec tématu pomocí metody `MqttClient.getTopic(java.lang.String topicString)`.

Filtry témat se používají k odběru témat a k přijímání publikování. Filtry témat mohou obsahovat zástupné znaky. Pomocí zástupných znaků 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 části Řetězec tématu. Syntaxe řetězců témat MQTT je popsána ve třídě `MqttClient` v dokumentaci rozhraní API pro MQTT client for Java. Odkazy na dokumentaci rozhraní API klienta pro knihovny klienta MQTT viz Referenční příručka programování klienta MQTT.

Syntaxe každého typu řetězce tématu je téměř identická. Existují čtyři drobné rozdíly:

1. Řetězce témat odeslané klientům IBM MQ uživateli MQTT musí odpovídat konvenci pro názvy správců front.
2. Maximální délky se liší. Délka řetězců tématu produktu IBM MQ je omezena na 10 240 znaků. Klient MQTT může vytvořit řetězce témat až do 65535 bajtů.
3. Řetězec tématu vytvořený klientem MQTT nemůže obsahovat znak null.
4. V produktu IBM Integration Busje úroveň tématu s hodnotou Null `'...//...'` neplatná. Úrovně tématu s hodnotou null jsou podporovány produktem IBM MQ.

Na rozdíl od IBM MQ publikování/odběru nemá protokol `mqttv3` koncepci objektu administrativního tématu. Z objektu tématu a řetězce tématu nelze vytvořit řetězec tématu. Řetězec tématu je však mapován na administrativní téma v souboru IBM MQ. Řízení přístupu přidružené k administrativnímu tématu určuje, zda je publikování publikováno v tématu, nebo zrušeno. Atributy, které jsou použity na publikování, když je předáno odběratelům, jsou ovlivněny atributy administrativního tématu.

Filtry témat

Syntaxe filtru témat IBM MQ je popsána v části Schéma zástupných znaků na základě tématu. Syntaxe filtrů témat, které můžete vytvořit pomocí klienta MQTT , je popsána ve třídě `MqttClient` v dokumentaci k rozhraní API pro MQTT client for Java. Odkazy na dokumentaci rozhraní API klienta pro knihovny klienta MQTT viz Referenční příručka programování klienta MQTT.

Související pojmy

Zpětná volání a synchronizace v klientských aplikacích MQTT
Programovací model klienta MQTT používá podprocesy ve velké míře. Podprocesy oddělují klientskou aplikaci systému MQTT v co největší míře od prodlev při přenosu zpráv na server a ze serveru. Publikování, tokeny doručení a události ztracených připojení jsou doručeny metodám ve třídě zpětného volání, která implementuje `MqttCallback`.

Vyčistit relace
Klient MQTT a služba telemetrie (MQXR) udržují informace o stavu relace. Informace o stavu se používají k zajištění "alespoň jednou" a "přesně jednou" doručení a "přesně jednou" přijetí publikování. 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 volby `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 můžete použít libovolný identifikační řetězec. Je důležité mít proceduru pro přidělení identifikátorů klientů a prostředky pro konfiguraci klienta s jeho zvoleným identifikátorem.

Tokeny doručení

Poslední vůle a vyhlášení závěti
Pokud připojení klienta MQTT neočekávaně skončí, můžete nakonfigurovat produkt MQ Telemetry tak, aby odeslal publikaci "last will and testament". Předem definujte obsah publikace a téma, na které má být publikace odeslána. "Poslední vůle a závěť" je vlastnost spojení. Vytvořte jej před připojením klienta.

Perzistence zpráv v klientech MQTT
Zprávy publikování jsou trvalé, pokud jsou odeslány s kvalitou služby "alespoň jednou"nebo "přesně jednou". Můžete implementovat vlastní mechanizmus perzistence na klientovi nebo použít výchozí mechanizmus perzistence, který je poskytován s klientem. Perzistence funguje v obou směrech pro publikace odeslané klientovi nebo z klienta.

Publikace
Publikace jsou instance produktu `MqttMessage` , které jsou přidruženy k řetězci tématu. Klienti systému MQTT mohou vytvářet publikování, která se odesílají do produktu IBM MQ, a odebírat témata týkající se produktu IBM MQ , která přijímají publikování.

Kvality služby poskytované klientem MQTT
Klient systému MQTT poskytuje tři kvality služeb pro doručování publikací produktu IBM MQ a klientu MQTT : "nejvýše jednou", "alespoň jednou" a "přesně jednou". Když klient MQTT odešle požadavek na vytvoření odběru do produktu IBM MQ , odešle se požadavek s kvalitou služby "alespoň jednou".

Zachovaná publikování a klienti MQTT
Téma může mít jedno a pouze jedno zachované publikování. Pokud vytvoříte odběr tématu, které má zachované publikování, publikování vám bude okamžitě předáno.

Odběry
Vytvořte odběry pro registraci zájmu o témata publikování pomocí filtru témat. Klient může vytvořit více odběrů nebo odběr obsahující filtr témat, který používá zástupné znaky k registraci zájmu o více témat. Publikace o tématech odpovídajících filtrům jsou odesílány klientovi. Odběry mohou zůstat aktivní po dobu, kdy je klient odpojen. Publikování jsou odesílána klientovi při opětovném připojení.

Vývoj aplikací Microsoft Windows Communication Foundation pomocí IBM MQ

Vlastní kanál produktu Microsoft Windows Communication Foundation (WCF) pro produkt IBM MQ odesílá a přijímá zprávy mezi klienty a službami WCF.

Související pojmy

“Úvod do vlastního kanálu IBM MQ pro WCF s produktem .NET” na stránce 1223
Vlastní kanál pro produkt IBM MQ je transportní kanál používající unifikovaný programovací model Microsoft Windows Communication Foundation (WCF).

“Použití vlastních kanálů IBM MQ pro WCF” na stránce 1227
Přehled informací, které jsou k dispozici pro programátory používající IBM MQ vlastní kanály pro Windows Communication Foundation (WCF).

“Použití ukázek WCF” na stránce 1245
Ukázky Windows Communication Foundation (WCF) poskytují několik jednoduchých příkladů, jak lze použít vlastní kanál IBM MQ .

FFST: Technologie podpory technologie WCF XMS First Failure

Související úlohy

Trasování vlastního kanálu WCF pro produkt IBM MQ

Odstraňování problémů s vlastním kanálem WCF pro produkt IBM MQ

Úvod do vlastního kanálu IBM MQ pro WCF s produktem .NET

Vlastní kanál pro produkt IBM MQ je transportní kanál používající unifikovaný programovací model Microsoft Windows Communication Foundation (WCF).

Rámec Microsoft Windows Communication Foundation, zavedený v produktu Microsoft.NET 3, umožňuje vývoj aplikací a služeb .NET nezávisle na přenosu a protokolech použitých k jejich připojení, což umožňuje použití alternativních přenosů nebo konfigurací v závislosti na prostředí, ve kterém je služba nebo aplikace implementována.

Připojení jsou spravována za běhu pomocí WCF sestavením zásobníku kanálu obsahujícího požadovanou kombinaci:

- Prvky protokolu: Volitelná sada prvků, kde lze přidat žádný, jeden nebo více prvků pro podporu protokolů, jako je například standard WS-*.
- Kodér zpráv: povinný prvek v zásobníku, který řídí serializaci zprávy do jejího formátu spojení.
- Transportní kanál: povinný prvek v zásobníku, který je zodpovědný za přenos serializované zprávy do koncového bodu.

Vlastní kanál pro produkt IBM MQ je transportním kanálem a jako takový musí být spárován s kódovačem zpráv a volitelnými protokoly, které aplikace vyžaduje s použitím vlastní vazby WCF. Tímto způsobem mohou aplikace, které byly vyvinuty pro použití WCF, používat vlastní kanál pro produkt IBM MQ k odesílání a přijímání dat stejným způsobem, jako používají vestavěné přenosy poskytované produktem Microsoft, což umožňuje jednoduchou integraci s asynchronními, rozšiřitelnými a spolehlivými funkcemi systému zpráv produktu IBM MQ. Úplný seznam podporovaných funkcí viz: “Funkce a možnosti vlastního kanálu WCF” na stránce 1227.

Kdy a proč mám používat vlastní kanál IBM MQ pro WCF?

Pomocí vlastního kanálu IBM MQ můžete odesílat a přijímat zprávy mezi klienty WCF a službami stejným způsobem jako vestavěné přenosy poskytované produktem Microsoft, což aplikacím umožňuje přístup k funkcím produktu IBM MQ v rámci unifikovaného programovacího modelu WCF.

Typické scénáře vzoru použití pro vlastní kanál IBM MQ pro WCF je jako rozhraní jiného než SOAP pro přenos nativních zpráv IBM MQ .

Zprávy přenášené pomocí jiného formátu než SOAP/Non-JMS (Pure MQMessage)

Používáte-li vlastní kanál IBM MQ pro WCF jako jiné rozhraní než SOAP pro přenos nativních zpráv IBM MQ , jsou zprávy přenášeny pomocí formátu zprávy jiného typu než SOAP/Non-JMS (Pure MQMessage) IBM MQ.

Uživatelé služby WCF mohou službu spustit, nebo jinými slovy, uživatelé služby mohou odeslat zprávu do fronty IBM MQ pomocí MQMessages. Aplikace mohou získat a nastavit pole MQMD a informační obsah. Je-li zpráva k dispozici ve frontách systému IBM MQ , může ji zpracovat libovolná služba WCF nebo jiné aplikace než WCF, například aplikace C nebo Java , které jsou spuštěny v systémech AIX, Linux, Windowsnebo z/OS.

Požadavky na software pro vlastní kanál IBM MQ pro WCF

Toto téma popisuje požadavky na software pro vlastní kanál IBM MQ pro WCF. Vlastní kanál IBM MQ pro WCF se může připojovat pouze ke správcům front IBM WebSphere MQ 7.0 nebo vyšším.

Požadavky na běhové prostředí

- Microsoft.NET Framework v4.7.2 nebo vyšší musí být nainstalován na hostitelském počítači.
- Produkt *Java a .NET Messaging and Web Services* je standardně nainstalován jako součást instalačního programu IBM MQ . Tato komponenta instaluje sestavení .NET potřebná pro vlastní kanál do mezipaměti Global Assembly Cache.

Poznámka: Pokud není produkt Microsoft .NET Framework V4.7.2 nebo vyšší nainstalován před instalací produktu IBM MQ, instalace produktu IBM MQ pokračuje bez chyb, ale produkt IBM MQ classes for .NET není k dispozici. Pokud je.NET Framework nainstalován po instalaci produktu IBM MQ, musí být sestavení produktu IBM MQ.NET registrována spuštěním skriptu *WMQInstallDir*\bin\amqiRegisterdotNet.cmd , kde *WMQInstallDir* je adresář, kde je nainstalován produkt IBM MQ . Tento skript nainstaluje požadovaná sestavení do mezipaměti GAC (Global Assembly Cache). Sada souborů amqi*.log , které zaznamenávají provedené akce, jsou vytvořeny v adresáři %TEMP% . Skript amqiRegisterdotNet.cmd není nutné znovu spustit, pokud je produkt .NET upgradován na verzi V4.7.2 nebo vyšší z dřívější verze, například z verze .NET V3.5.

Požadavky na vývojové prostředí

- Microsoft Visual Studio 2015 nebo Windows Software Development Kit for .NET 4.7.2 nebo novější.
- Microsoft.NET Framework V4.7.2 nebo vyšší musí být nainstalován na hostitelském počítači, aby bylo možné sestavit ukázkové soubory řešení.

IBM MQ vlastní kanál pro WCF: Co je nainstalováno?

Vlastní kanál pro produkt IBM MQ je transportní kanál používající unifikovaný programovací model Microsoft Windows Communication Foundation (WCF). Vlastní kanál je standardně nainstalován jako součást instalace.

IBM MQ vlastní kanál pro WCF

Vlastní kanál a jeho závislosti jsou obsaženy v komponentě Java and .NET Messaging and Web Services , která je standardně nainstalována. Při upgradu produktu IBM MQ z dřívější verze nežIBM MQ 8.0aktualizace standardně nainstaluje vlastní kanál IBM MQ pro WCF, pokud byla komponenta Java and .NET Messaging and Web Services 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 hlavním vlastním sestavením kanálu, které obsahuje třídy rozhraní WCF. Tyto soubory jsou nainstalovány v globální mezipaměti sestavení (GAC) a jsou také k dispozici v následujícím adresáři: *MQ_INSTALLATION_PATH* \bin , kde *MQ_INSTALLATION_PATH* je adresář, ve kterém je nainstalován produkt IBM MQ .

Následující tabulka shrnuje třídy klíčů, které jsou nezbytné pro použití vlastního kanálu.

Tabulka 189. Klíčové třídy nezbytné pro použití vlastního kanálu

	Rozhraní SOAP/JMS (existující)	**Jiné rozhraní než SOAP/Non-JMS (od IBM MQ 8.0)**
Sestavení vlastního kanálu	IBM.XMS.WCF.dll	IBM.WMQ.WCF.dll
Název vazby přenosu	IBM.XMS.WCF.SoapJmsIbmTransportBindingElement	IBM.WMQ.WCF.WmqIbmTransportBindingElement
Importér transportních vazeb	IBM.XMS.WCF.SoapJmsIbmTransportBindingElementImporter	IBM.WMQ.WCF.WmqIbmTransportBindingElementImporter
Konfigurace vazby přenosu	IBM.XMS.WCF.SoapJmsIbmTransportBindingElementConfig	IBM.WMQ.WCF.WmqIbmTransportBindingElementConfig
Ukázky (jednosměrné)	SimpleOneWay_Client, SimpleOneWay_Service	MQMessaging_OneWay_Client, MQMessaging_OneWay_Service
Ukázky (RequestReply)	SimpleRequestReply_Client, SimpleRequestReply_Service	MQMessaging_RequestReply_Client, MQMessaging_RequestReply_Service

IBM.WMQ.WCF.dll podporuje rozhraní SOAP/JMS i jiná než SOAP/Non-JMS . Doporučuje se, aby nové vyvinuté aplikace používaly IBM.WMQ.WCF , protože podporuje obě rozhraní.

Odesílání formátovaných zpráv MQSTR

Pokud je zpráva požadavku typu MQSTR, můžete vybrat odeslání zprávy odpovědi ve formátu MQSTR.

Chcete-li změnit formát zprávy odpovědi, musíte použít další parametr identifikátoru URI **replyMessageFormat** . Podporované hodnoty jsou:

""

"" " je výchozí hodnota.

Zpráva odpovědi je ve formátu bajtů (MQMFT_NONE). Příklad:

```
"jms:/queue?
destination=SampleQ@QM1&connectionFactory=binding(server)connectQueueManager(QM1)
&initialContextFactory=com.ibm.mq.jms.Nojndi&replyDestination=SampleReplyQ&replyMessageForma
t= "
```

MQSTR

Zpráva odpovědi je 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. V hodnotě parametru **replyMessageFormat** se nerozlišují velká a malá písmena.
2. Použití jiné hodnoty než "" nebo *MQSTR*způsobí výjimku neplatné hodnoty parametru.

IBM MQ vlastní ukázky kanálů

Ukázky poskytují několik jednoduchých příkladů, jak lze použít vlastní kanál IBM MQ pro 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 IBM MQ viz “Použití ukázek WCF” na stránce 1245.

svcutil.exe.config

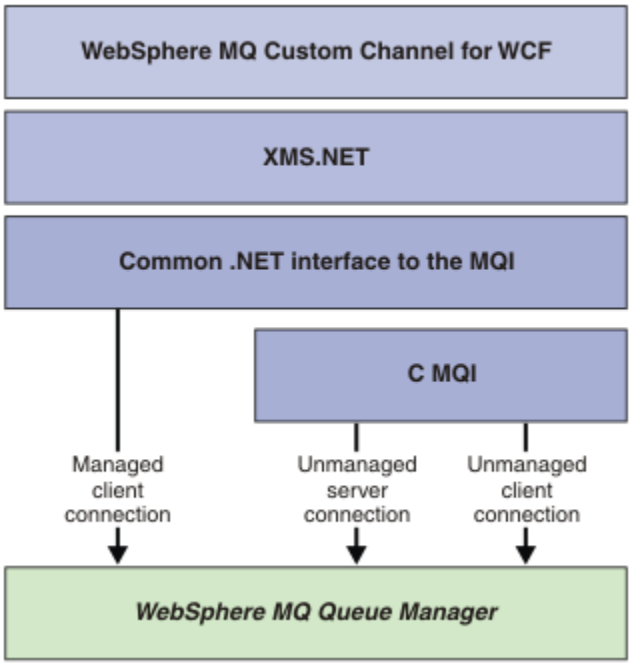
`svcutil.exe.config` je příkladem nastavení konfigurace, která jsou nezbytná k tomu, aby nástroj pro generování serveru proxy klienta Microsoft WCF `svcutil` mohl rozpoznat vlastní kanál. Soubor `svcutil.exe.config` se nachází v adresáři *MQ_INSTALLATION_PATH* `\tools\wcf\docs\examples\` , kde *MQ_INSTALLATION_PATH* je instalační adresář pro IBM MQ. Další informace o použití konzoly `svcutil.exe.config`viz “Generování konfiguračních souborů serveru proxy klienta WCF a aplikace pomocí nástroje svcutil s metadaty ze spuštěné služby” na stránce 1243.

Architektura WCF

Vlastní kanál IBM MQ pro WCF je integrován v horní části rozhraní API produktu `IBM Message Service Client for .NET (XMS .NET)` .

Rozhraní SOAP/JMS

Architektura WCF je znázorněno na následujícím diagramu:



Obrázek 149. Architektura WCF pro rozhraní SOAP/JMS

Všechny požadované komponenty se standardně instalují s instalací produktu.

Tato tři připojení jsou:

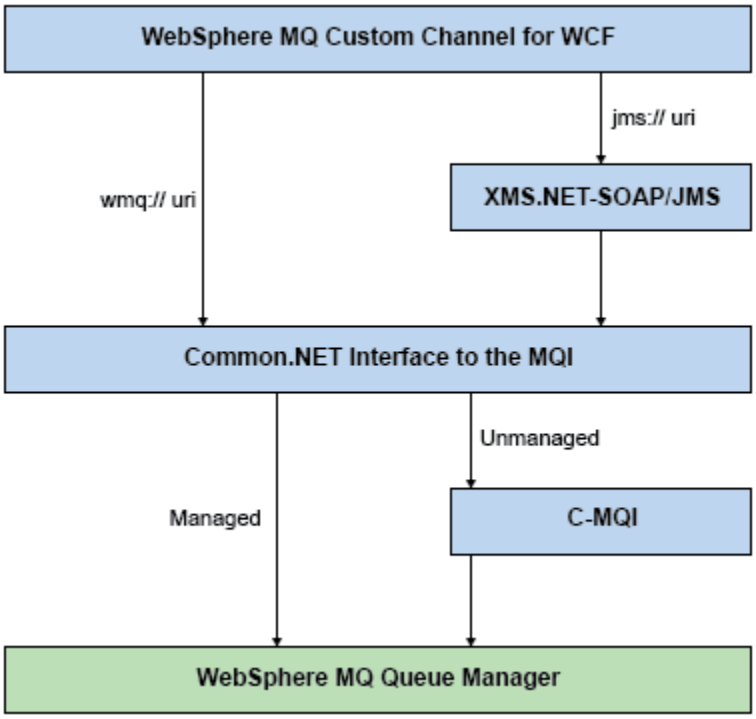
- Připojení spravovaných klientů
- Nespravovaná připojení serveru
- Nespravovaná připojení klienta

Další informace o těchto připojeních viz “Volby připojení WCF” na stránce 1233.

Jiné rozhraní než SOAP/Non-JMS

Vlastní kanál IBM MQ pro službu WCF podporuje rozhraní SOAP/JMS (k dispozici v produktu IBM WebSphere MQ 7.0.1). a rozhraní Non-SOAP/Non-JMS .

Architektura WCF je znázorněno na následujícím diagramu:



Obrázek 150. Architektura WCF pro rozhraní Non-SOAP/Non-JMS

Použití vlastních kanálů IBM MQ pro WCF

Přehled informací, které jsou k dispozici pro programátory používající IBM MQ vlastní kanály pro Windows Communication Foundation (WCF).

Produkt Microsoft Windows Communication Foundation podporuje webové služby a podporu systému zpráv v rámci Microsoft.NET Framework 3. IBM MQ lze použít jako vlastní kanál v rámci WCF v rámci .NET Framework 3 stejným způsobem jako vestavěné kanály nabízené produktem Microsoft.

Zprávy přenášené přes vlastní kanál jsou formátovány podle protokolu SOAP prostřednictvím JMS implementace IBM MQ. Aplikace pak mohou komunikovat se službami, jejichž hostitelem je WCF nebo infrastruktura služeb WebSphere SOAP over JMS .

Funkce a možnosti vlastního kanálu WCF

Informace týkající se funkcí a možností vlastního kanálu WCF naleznete v následujících tématech.

Tvary vlastních kanálů WCF

Přehled vlastních tvarů kanálů, které lze IBM MQ použít v rámci vlastních kanálů produktu Microsoft Windows Communication Foundation (WCF).

Vlastní kanál IBM MQ pro WCF podporuje dva tvary kanálů:

- Jednosměrná
- Požadavek-odpověď

WCF automaticky vybere tvar kanálu podle hostované servisní smlouvy.

Smlouvy, které obsahují metody používající pouze parametr **IsOneWay** , jsou obsluhovány tvarem jednosměrného kanálu, například:

```
[OperationContract(IsOneWay = true)]
void printString(String text);
```

Smlouvy, které zahrnují buď kombinaci jednosměrných metod a metod typu požadavek-odezva, nebo všechny metody typu požadavek-odezva, jsou obsluhovány tvarem kanálu požadavek-odezva. Příklad:

```
[OperationContract]
int subtract(int a, int b);

[OperationContract(IsOneWay = true)]
void printString(string text);
```

Poznámka: Při kombinování jednosměrných metod a metod typu požadavek-odezva ve stejné smlouvě musíte zajistit, aby bylo chování stejné, zejména při práci ve smíšeném prostředí, protože jednosměrné metody čekají, až od služby obdrží odpověď s hodnotou null.

Jednosměrný kanál

Jednosměrný vlastní kanál IBM MQ pro WCF se používá například k odesílání zpráv z klienta WCF pomocí tvaru jednosměrného kanálu. Kanál může odesílat zprávy pouze v jednom směru, například ze správce front klienta do fronty ve službě WCF.

Kanál požadavek-odpověď

Vlastní kanál IBM MQ požadavek-odezva pro WCF se používá například k asynchronnímu odesílání zpráv ve dvou směrech. Stejná instance klienta musí být použita pro asynchronní zasílání zpráv. Kanál může odesílat zprávy v jednom směru, například ze správce front klienta do fronty ve službě WCF, a poté odesílat zprávu odpovědi z WCF do fronty ve správci front klienta.

Názvy a hodnoty parametrů identifikátoru URI WCF

Názvy a hodnoty parametrů URI pro rozhraní SOAP/JMS a jiné než SOAP/Non JMS .

Rozhraní SOAP/JMS

connectionFactory
Parametr connectionFactory je povinný.

InitialContextFactory
Parametr továrny initialContextje povinný a musí být nastaven na "com.ibm.mq.jms.Nojndi" pro kompatibilitu s produktem WebSphere Application Server a dalšími produkty.

Jiné rozhraní než SOAP/Non JMS

Formát identifikátoru URI je stejný jako pro specifikace MA93 . Další podrobnosti o IBM MQ specifikacích 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 Příklad IRI
Následující příklad IRI říká klientovi služby, že může použít připojení IBM MQ vazby klienta TCP k počítači s názvem example.com na portu 1414 a vložit zprávy trvalých požadavků do fronty

s názvem SampleQ ve správci front QM1. Hodnota IRI určuje, že poskytovatel služeb vloží odpovědi do fronty s názvem SampleReplyQ.

```
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 vytvořit zabezpečená připojení (TLS) pomocí klienta/služby WCF, nastavte následující vlastnosti s příslušnými hodnotami v identifikátoru URI. Všechny vlastnosti s předponou "*" jsou povinné pro vytvoření zabezpečeného připojení.

- **sslKeyRepository**: *SYSTEM nebo *USER
- * **sslCipherSpec**: platná 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"
```

WCF vlastní kanál zajištěné doručení

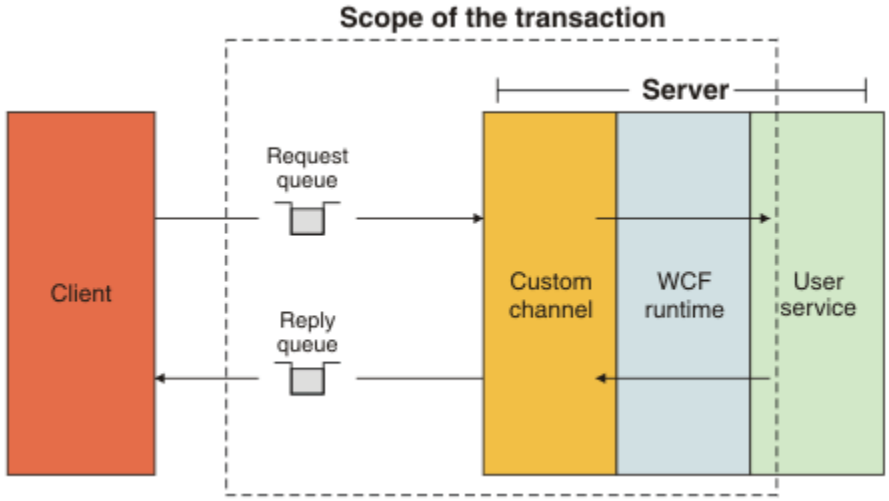
Zajištěné doručení zaručuje, že požadavek na službu nebo odpověď je provedeno a není ztraceno.

Je přijata zpráva požadavku a jakákoli zpráva odpovědi je odeslána v rámci synchronizačního bodu lokální transakce, který lze odvolat v případě selhání běhového prostředí. Příklady těchto selhání: Neošetřená výjimka vyvolaná službou, selhání odbavení zprávy do služby nebo selhání doručení zprávy odpovědi.

`AssuredDelivery` je atribut zajištěného doručení, který lze zadat ve smlouvě o službách, aby bylo zaručeno, že všechny zprávy požadavků přijaté službou a všechny zprávy odpovědí odeslané ze služby nebudou ztraceny v případě selhání běhového prostředí.

Chcete-li zajistit, aby byly zprávy zachovány i v případě selhání systému nebo výpadku napájení, musí být zprávy odeslány jako trvalé. Chcete-li používat trvalé zprávy, musí mít aplikace klienta tuto volbu určenou v identifikátoru URI koncového bodu.

Distribuované transakce nejsou podporovány a rozsah transakce nepřekračuje rámec zpracování zprávy požadavku a odpovědi provedené produktem IBM MQ. Jakákoli práce provedená v rámci služby může být znovu spouštěna v důsledku selhání, které způsobí, že zpráva bude znovu přijata. Následující diagram zobrazuje rozsah transakce:



Zajištěné doručení je povoleno použitím atributu `AssuredDelivery` na třídu služeb, jak ukazuje následující příklad:

```
[AssuredDelivery]
class TestCalculatorService : IWMQSampleCalculatorContract
{
    public int add(int a, int b)
    {
        int ans = a + b;
        return ans;
    }
}
```

Při použití atributu `AssuredDelivery` musíte mít na paměti následující body:

- Když kanál zjistí, že v případě odvolání a opětného přijetí zprávy dojde pravděpodobně k selhání, bude zpráva považována za nezpracovatelnou a nebude vrácena do fronty požadavků pro opětovné zpracování. Například: Pokud přijatá zpráva není správně naformátována nebo ji nelze odbavit do služby. Neošetřené výjimky vyvolané operací služby jsou vždy znovu odeslány, dokud nebyla zpráva znovu doručena, a to v maximálním počtu, který určuje vlastnost prahové hodnoty vrácení fronty požadavků. Další informace viz: “Nezpracovatelné zprávy vlastního kanálu WCF” na stránce 1231
- Kanál provádí čtení, zpracování a odpovídání na každou zprávu požadavku jako atomickou operaci s použitím jednoho podprocesu provedení k vynucení transakční integrity. Chcete-li povolit souběžné spouštění operací služby, kanál umožňuje službě 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 1239
- Funkce zajištěného doručení používá body rozšiřitelnosti IOperationInvoker a IErrorHandler WCF. Pokud jsou tyto body rozšiřitelnosti externě používány aplikací, musí aplikace zajistit, aby byly volány všechny dříve registrované body rozšiřitelnosti. Pokud tak neučiním pro IErrorHandler , může dojít k nehlášeným chybám. Pokud tak neučiníte pro IOperationInvoker , může to způsobit, že WCF přestane odpovídat.

Vlastní zabezpečení kanálu WCF

Vlastní kanál IBM MQ pro službu WCF podporuje použití protokolu TLS pouze pro nespravovaná připojení klienta ke správci front.

Zadejte protokol TLS pomocí položky v tabulce CCDT (Client Channel Definition Table). Další informace o tabulkách CCD viz Tabulka definic kanálů klienta.

Tabulky definic kanálů klienta WCF (CCDT)

Vlastní kanál IBM MQ pro WCF podporuje použití tabulek definic kanálů klienta (CCDT) ke konfiguraci informací o připojení pro připojení klienta.

Tabulky CDT jsou řízeny prostřednictvím těchto dvou proměnných prostředí:

- *MQCHLLIB* uvádí adresář, ve kterém je umístěna tabulka.
- *MQCHLTAB* uvádí název souboru tabulky.

Jsou-li tyto proměnné prostředí definovány, mají přednost před všemi podrobnostmi připojení klienta uvedenými v identifikátoru URI.

Další informace o tabulkách definic kanálů klienta viz: Tabulka definic kanálů klienta.

Nezpracovatelné zprávy vlastního kanálu WCF

Pokud se službě nepodaří zpracovat zprávu požadavku nebo nedoručí zprávu odpovědi do fronty odpovědí, je zpráva považována za nezpracovatelnou zprávu.

Nezpracovatelné zprávy požadavku

Pokud zprávu požadavku nelze zpracovat, bude považována za nezpracovatelnou zprávu. Tato akce zabrání službě v opětovném přijetí stejné nezpracovatelné zprávy. Aby byla nezpracovatelná zpráva požadavku považována za nezpracovatelnou zprávu, musí být splněna jedna z následujících situací:

- Počet vrácení zpráv překročil prahovou hodnotu vrácení uvedenou ve frontě požadavků, která se vyskytne pouze v případě, že bylo pro službu uvedeno zajištěné doručení. Další informace o zajištěném doručení naleznete na adrese: “WCF vlastní kanál zajištěné doručení” na stránce 1229
- Zpráva nebyla správně naformátována a nelze ji interpretovat jako zprávu SOAP přes JMS .

Nezpracovatelné zprávy odpovědí

Pokud se službě nepodaří doručit zprávu odpovědi do fronty odpovědí, je zpráva odpovědi považována za nezpracovatelnou zprávu. V případě zpráv s odpovědí tato akce umožňuje pozdější načtení zpráv s odpovědí, což pomáhá při určování problémů.

Zpracování nezpracovatelných zpráv

Akce prováděná pro nezpracovatelnou zprávu závisí na konfiguraci správce front a na hodnotách nastavených ve volbách sestavy zprávy. Pro protokol SOAP přes produkt JMSjsou standardně nastaveny následující volby sestavy pro zprávy požadavků a nelze je konfigurovat:

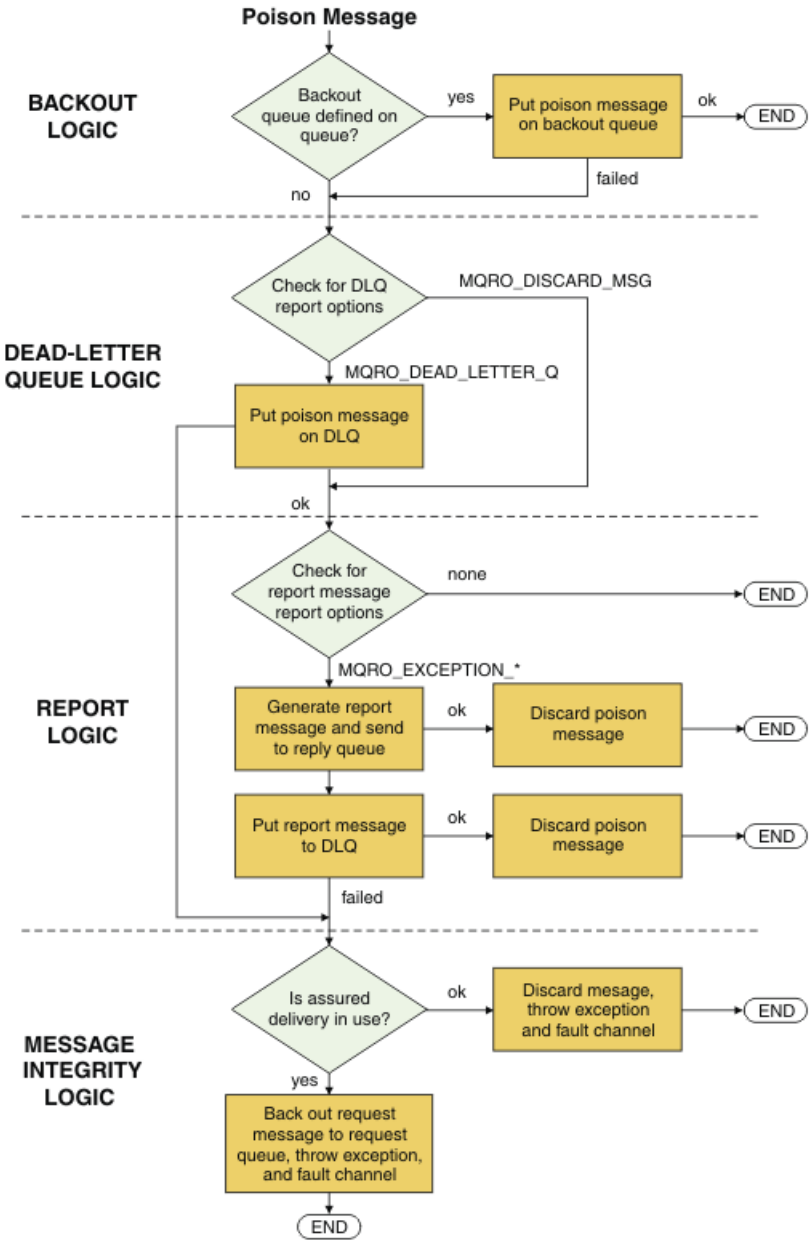
- MQRO_EXCEPTION_WITH_FULL_DATA
- MQRO_EXPIRATION_WITH_FULL_DATA
- MQRO_DISCARD_MSG

Pro SOAP přes JMSje standardně nastavena následující volba sestavy pro zprávy odpovědí a není konfigurovatelná:

- MQRO_DEAD_LETTER_Q

Pokud zprávy pocházejí ze zdroje jiného než WCF, podívejte se do dokumentace pro tento zdroj.

Následující diagram zobrazuje možné akce a kroky provedené v případě, že zpracování nezpracovatelných zpráv selže:



Schopnosti zpráv IBM MQ pro aplikace WCF

Jiné než SOAP/Non-JMS (tj. IBM MQ) schopnosti zpráv pro aplikace WCF.

Pro rozhraní non-SOAP/Non-JMS jsou schopnosti zpráv IBM MQ pro aplikace WCF následující:

- Aplikace WCF mohou odesílat a přijímat základní zprávy IBM MQ , které mohou být zpracovány libovolnou aplikací IBM MQ .
- Aplikace WCF mají úplnou kontrolu nad aktualizací MQMD a informačního obsahu.
- Klient WCF může odesílat zprávy systému IBM MQ , které mohou využívat klienti systému IBM MQ , například klienti C, Java, JMSa .NET .

Rozhraní WCF pro non-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 Bajty
- WmqXmlZpráva pro informační obsah typu XML

Chcete-li nastavit informační obsah zprávy, použijte v závislosti na typu informačního obsahu vlastnost **Data** pro třídu zpráv WmqString, WmqBytesMessage nebo WmqXml. Například použijte následující kód k nastavení informačního obsahu typu String:

```
WmqStringMessage strMsg = new WmqStringMessage();
//Setting the Message PayLoad
strMsg.Data = "Hello World";
//MQMD property
strMsg.Format = WmqMessageFormat.MQFMT_STRING;
```

Volby připojení WCF

Existují tři režimy připojení vlastního kanálu IBM MQ pro WCF ke správci front. Zvažte, který typ připojení nejlépe vyhovuje vašim požadavkům.

Další informace o volbách připojení viz: “Rozdíly v připojení” na stránce 563

Další informace o architektuře WCF viz: “Architektura WCF” na stránce 1225

Nespravované připojení klienta

Připojení provedené v tomto režimu se připojí 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 nainstalovat pomocí konzoly IBM MQ MQI clientbuď na serveru IBM MQ , nebo na samostatném počítači.

Nespravované připojení k serveru

Při použití v režimu vazeb serveru používá vlastní kanál IBM MQ pro prostředek WCF rozhraní API správce front namísto komunikace prostřednictvím sítě. Použití připojení vazeb poskytuje lepší výkon pro aplikace IBM MQ než použití síťových připojení.

Chcete-li použít připojení vazeb, musíte nainstalovat vlastní kanál IBM MQ pro WCF na serveru IBM MQ .

Připojení spravovaného klienta

Připojení provedené v tomto režimu se připojí jako klient IBM MQ k serveru IBM MQ spuštěnému buď na lokálním počítači, nebo na vzdáleném počítači.

IBM MQ Vlastní třídy kanálu pro připojení .NET 3 v tomto režimu zůstávají ve spravovaném kódu .NET a neprovádějí žádná volání nativních služeb. Další informace o spravovaném kódu naleznete v dokumentaci k produktu Microsoft .

Použití spravovaného klienta má řadu omezení. Další informace o těchto omezeních viz “Připojení spravovaných klientů” na stránce 563.

Vytvoření a konfigurace vlastního kanálu IBM MQ pro WCF

Vlastní kanály IBM MQ pro WCF pracují stejným způsobem jako kanály WCF přenosu nabízené produktem Microsoft. Vlastní kanál IBM MQ pro WCF lze vytvořit jedním ze dvou způsobů.

Informace o této úloze

Vlastní kanál IBM MQ je integrován s kanálem WCF jako transportním kanálem WCF a jako takový musí být spárován s kodérem zpráv a volitelnými kanály protokolu, aby mohl vytvořit úplný zásobník kanálů, který může být používán aplikací. Pro úspěšné vytvoření kompletního zásobníku kanálů jsou vyžadovány dva prvky:

1. Definice vazby: Určuje, které prvky jsou nezbytné pro sestavení zásobníku aplikačních kanálů, včetně transportního kanálu, kodéru zpráv a libovolných protokolů, a dále všechna obecná nastavení konfigurace. Pro vlastní kanál musí být definice vazby vytvořena ve formě vlastní vazby WCF.

2. Definice koncového bodu: Propojí smlouvu o službě s definicí vazby a také poskytne skutečný identifikátor URI připojení, který popisuje, kde se může aplikace připojit. Pro vlastní kanál je identifikátor URI ve formě protokolu SOAP prostřednictvím identifikátoru URI JMS .

Tyto definice lze vytvořit jedním ze dvou různých způsobů:

- Administrativně; Definice jsou vytvořeny poskytnutím podrobností v konfiguračním souboru aplikace (například: `app.config`).
- Programově; Definice jsou vytvořeny přímo z kódu aplikace.

Rozhodnutí, kterou metodu použít k vytvoření definic, musí být založeno na požadavcích aplikace, jak je uvedeno níže:

- Administrativní metoda pro konfiguraci poskytuje flexibilitu při změně podrobností služby a klienta 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 kanálu WCF administrativně poskytnutím informací o vazbách a koncových bodech v konfiguračním souboru aplikace

Vlastní kanál IBM MQ pro WCF je kanálem WCF na úrovni transportu. Koncový bod a vazba musí být definovány tak, aby používaly vlastní kanál, a tyto definice lze provést dodáním informací o vazbách a koncových bodech v konfiguračním souboru aplikace.

Chcete-li konfigurovat a používat vlastní kanál IBM MQ pro prostředek WCF, který je kanálem WCF na úrovni transportu, musí být definována vazba a definice koncového bodu. Vazba uchovává informace o konfiguraci kanálu a definice koncového bodu uchovává podrobnosti o připojení. Tyto definice lze vytvořit dvěma způsoby:

- Programově přímo z kódu aplikace, jak je popsáno zde: "Vytvoření vlastního kanálu WCF programovým potlačením informací o vazbách a koncových bodech" na stránce 1236
- Administrativně poskytnutím podrobností v konfiguračním souboru aplikace, jak je popsáno v následujícím postupu.

Konfigurační soubor aplikace klienta nebo služby se obvykle nazývá *yourappname*`.exe.config` , kde *název_aplikace* je název vaší aplikace. Konfigurační soubor aplikace lze snadno upravit pomocí nástroje editoru konfigurací služeb Microsoft s názvem `SvcConfigEditor.exe` následujícím způsobem:

- Spusťte nástroj editoru konfigurací `SvcConfigEditor.exe` . Výchozí umístění instalace nástroje je: *`Drive:`*`\Program Files\Microsoft SDKs\Windows\v6.0\Bin\SvcConfigEditor.exe` , kde *Jednotka:* je název instalační jednotky.

Krok 1: Přidejte rozšíření prvku vazby, které umožní WCF vyhledat vlastní kanál

1. Klepnutím pravým tlačítkem myši na volbu **Rozšířené** > **Rozšíření** > **prvek vazby** otevřete nabídku a vyberte volbu **Nový** .
2. Vyplňte pole, jak je uvedeno v této tabulce:

Tabulka 190. Nová pole prvku vazby

Pole	**Hodnota**
Název	`IBM.XMS.WCF.SoapJmsIbmTransportChannel`
Typ	Přejděte do adresáře `IBM.XMS.WCF.dll` v globální mezipaměti sestavení (GAC) a vyberte volbu `IBM.XMS.WCFSoapJmsIbmTransportBindingElementConfig`

Krok 2: Vytvořte vlastní definici vazby, která bude párovat vlastní kanál s kódovačem 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 191. Nová pole konfigurace vazby	
Pole	**Hodnota**
Název	`CustomBinding_WMQ`
BindingElement 1	`textMessageEncoding (MessageVersion: Soap11)`
BindingElement 2	`IBM.XMS.WCF.SoapJmsIbmTransportChannel`

Krok 3: Určení vlastností vazby

1. Vyberte *IBM.XMS.WCF.SoapJmsIbmTransportChannel* z vazby, kterou jste vytvořili v: “Krok 2: Vytvořte vlastní definici vazby, která bude párovat vlastní kanál s kódovačem zpráv WCF” na stránce 1235
2. Proveďte požadované změny výchozích hodnot vlastností, jak je popsáno v tématu: “Volby konfigurace vazby WCF” na stránce 1239

Krok 4: Vytvoření definice koncového bodu

Vytvořte definici koncového bodu, která odkazuje na vlastní vazbu, kterou jste vytvořili v: “Krok 2: Vytvořte vlastní definici vazby, která bude párovat vlastní kanál s kódovačem zpráv WCF” na stránce 1235 , a poskytuje podrobnosti o připojení služby. Způsob, jakým jsou tyto informace zadány, závisí na tom, zda je definice určena pro klientskou aplikaci nebo aplikaci služeb.

V případě klientské aplikace přidejte definici koncového bodu do klientské sekce následujícím způsobem:

1. Klepnutím pravým tlačítkem myši na volbu **Klient** > **Koncové body** otevřete nabídku a vyberte volbu **Nový koncový bod klienta** .
2. Vyplňte pole, jak je uvedeno v této tabulce:

Tabulka 192. Nová pole koncového bodu klienta	
Pole	**Hodnota**
Název	`Endpoint_WMQ`
Adresa	*Identifikátor URI SOAP/JMS popisující podrobnosti připojení WMQ potřebné pro přístup ke službě.* Další podrobnosti viz: “IBM MQ vlastní kanál pro formát adresy URI koncového bodu WCF” na stránce 1237
Vazba	`customBinding`
BindingConfiguration	`CustomBinding_WMQ`
Smlouva.	*Název rozhraní servisní smlouvy*

V případě aplikace služeb přidejte definici služby do sekce služeb takto:

1. Klepnutím pravým tlačítkem myši na položku **Služby** otevřete nabídku a vyberte volbu **Nová služba**, poté vyberte třídu služeb, která má být hostována.
2. Přidejte definici koncového bodu do sekce **Koncové body** pro novou službu a vyplňte pole, jak je uvedeno v této tabulce:

Tabulka 193. Nová pole koncového bodu služby	
Pole	**Hodnota**
Název	`Endpoint_WMQ`
Adresa	*Identifikátor URI SOAP/JMS popisující podrobnosti připojení WMQ potřebné pro přístup ke službě.* Další podrobnosti viz: “IBM MQ vlastní kanál pro formát adresy URI koncového bodu WCF” na stránce 1237
Vazba	`customBinding`
BindingConfiguration	`CustomBinding_WMQ`
Smlouva.	*Název implementační třídy služeb*

Vytvoření vlastního kanálu WCF programovým potlačením informací o vazbách a koncových bodech

Vlastní kanál IBM MQ pro WCF je kanálem WCF na úrovni transportu. Koncový bod a vazba musí být definovány tak, aby používaly vlastní kanál, a tyto definice lze provádět programově přímo z kódu aplikace.

Chcete-li konfigurovat a používat vlastní kanál IBM MQ pro prostředek WCF, který je kanálem WCF na úrovni transportu, musí být definována vazba a definice koncového bodu. Vazba uchovává informace o konfiguraci kanálu a definice koncového bodu uchovává podrobnosti o připojení. Další informace viz “Použití ukázek WCF” na stránce 1245.

Tyto definice lze vytvořit dvěma způsoby:

- Administrativně poskytnutím podrobností v konfiguračním souboru aplikace, jak je popsáno v tématu “Vytvoření vlastního kanálu WCF administrativně poskytnutím informací o vazbách a koncových bodech v konfiguračním souboru aplikace” na stránce 1234.
- Programově přímo z kódu aplikace, jak je popsáno v následujících dílčích tématech.

Programové definování informací o vazbách a koncových bodech: rozhraní SOAP/JMS

Pro rozhraní SOAP/JMS můžete definovat koncový bod a vazbu programově přímo z kódu aplikace.

Informace o této úloze

Chcete-li dodat informace o vazbách a koncových bodech programově, přidejte požadovaný kód do aplikace provedením následujících kroků.

Postup

1. Vytvořte instanci prvku vazby přenosu kanálu přidáním následujícího kódu do aplikace:

```
SoapJmsIbmTransportBindingElement transportBindingElement = new
SoapJmsIbmTransportBindingElement();
```

2. Nastavte všechny požadované vlastnosti vazby, například přidáním následujícího kódu do aplikace, abyste nastavili `ClientConnectionMode`:

```
transportBindingElement.ClientConnectionMode = XmsWCFBindingProperty.AS_URI;
```

3. Vytvořte vlastní vazbu, která bude párovat transportní kanál s kódovačem zpráv, a to přidáním následujícího kódu do aplikace:

```
Binding binding = new CustomBinding(new TextMessageEncodingBindingElement(),
transportBindingElement);
```

4. Vytvořte identifikátor URI SOAP/JMS .

   Identifikátor URI SOAP/JMS , který popisuje podrobnosti připojení IBM MQ požadované pro přístup ke službě, musí být zadán jako adresa koncového bodu. Zadaná adresa závisí na tom, zda je kanál používán pro aplikaci služeb nebo klientskou aplikaci.

   - Pro klientské aplikace musí být identifikátor URI SOAP/JMS vytvořen jako `EndpointAddress` následujícím způsobem:

     ```
     EndpointAddress address = new EndpointAddress("jms:/queue?
     destination=SampleQ@QM1&connectionFactory
     =connectQueueManager(QM1)&initialContextFactory=com.ibm.mq.jms.Nojndi");
     ```

   - V případě aplikací služeb musí být identifikátor URI SOAP/JMS vytvořen jako identifikátor URI takto:

     ```
     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 “IBM MQ vlastní kanál pro formát adresy URI koncového bodu WCF” na stránce 1237.

Programové definování informací o vazbách a koncových bodech: rozhraní non-SOAP/Non-JMS
Pro 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 vazbách a koncových bodech programově, přidejte požadovaný kód do aplikace provedením následujících kroků.

Postup

1. Vytvořte WmqBinding přidáním následujícího kódu do aplikace:

   ```
   WmqBinding binding = new WmqBinding();
   ```

   Tento kód vytváří vazbu, která spojuje prvky WmqMsgEncodingElement a WmqIbmTransportBindingnezbytné pro rozhraní Non-SOAP/Non-JMS .
2. Zadejte identifikátor URI wmq: //, který popisuje podrobnosti připojení IBM MQ nezbytné pro přístup ke službě.

   Způsob, jakým zadáte identifikátor URI wmq: //, závisí na tom, zda je kanál používán pro aplikaci služeb nebo klientskou aplikaci.

   - Pro klientské aplikace musí být identifikátor URI wmq: // vytvořen jako EndpointAddress následujícím způsobem:

     ```
     EndpointAddress address = new EndpointAddress
     ("wmq://localhost:1414/msg/queue/Q1?connectQueueManager=QM1&replyTo=Q2");
     ```

   - V případě aplikací služeb musí být identifikátor URI wmq: // vytvořen jako identifikátor URI následujícím způsobem:

     ```
     Uri sampleAddress = new Uri(
     "wmq://localhost:1414/msg/queue/Q1?connectQueueManager=QM1&replyTo=Q2");
     ```

IBM MQ vlastní kanál pro formát adresy URI koncového bodu WCF

Webová služba je určena 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 jiné rozhraní než SOAP/Non-JMS .

Rozhraní SOAP/JMS

Formát identifikátoru URI, který je podporován v přenosu IBM MQ pro protokol SOAP, umožňuje komplexní stupeň kontroly nad parametry a volbami specifickými pro SOAP/ IBM MQ při přístupu k cílovým službám. Tento formát je kompatibilní s produktem WebSphere Application Server a s produktem CICS, což usnadňuje integraci produktu IBM MQ s oběma těmito produkty.

Syntaxe identifikátoru URI je následující:

```
jms:/queue? name=value&name=value...
```

kde name je název parametru a *hodnota* je odpovídající hodnota a prvek name = *value* lze libovolně opakovat, přičemž druhému a následnému výskytu předchází znak ampersand (&).

V názvech parametrů se rozlišují malá a velká písmena, stejně jako v názvech objektů IBM MQ . Je-li některý parametr zadán více než jednou, použije se konečný výskyt parametru, což znamená, že klientské aplikace mohou přepsat hodnoty parametrů připojením k identifikátoru URI. Jsou-li zahrnuty další nerozpoznané parametry, budou ignorovány.

Pokud uložíte identifikátor URI do řetězce XML, musíte znak ampersand reprezentovat jako "&". Podobně, pokud je identifikátor URI kódován ve skriptu, dbejte na změnu významu znaků, jako např. **&** , 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
```

Jsou zadány pouze požadované parametry (`targetService` je vyžadováno pouze pro služby .NET) a `connectionFactory` nemá k dispozici žádné volby.

V tomto příkladu osy `connectionFactory` obsahuje 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 uvedena volba `sslPeerName` `connectionFactory` . Samotná hodnota 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 při přístupu k cílovým službám komplexní stupeň řízení parametrů a voleb specifických pro produkt IBM MQ .

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
```

Toto IRI sděluje klientovi služby, že může použít připojení IBM MQ TCP client-binding connection to a machine s názvem example.com na portu 1415 a vložit zprávy trvalých požadavků do fronty s názvem

INS.QUOTE.REQUEST ve správci front MOTOR.INS. IRI uvádí, že poskytovatel služeb vkládá odpovědi do fronty s názvem INS.QUOTE.REPLY ve správci front BRANCH452. Formát identifikátoru URI je uveden pro SupportPac MA93. Další podrobnosti o specifikacích IBM MQ IRI viz SupportPac MA93: IBM MQ -Definice služby .

Volby konfigurace vazby WCF

Existují dva způsoby použití voleb konfigurace na informace o vazbách vlastních kanálů. Buď nastavíte vlastnosti administrativně, nebo je nastavíte programově.

Volby konfigurace vazby lze nastavit jedním ze dvou různých způsobů:

1. Administrativně: Nastavení vlastnosti vazby musí být určeno v sekci přenosu definice vlastní vazby v konfiguračním souboru aplikací, například: `app.config`.
2. Programově: Kód aplikace musí být upraven tak, aby určoval vlastnost během inicializace vlastní vazby.

Administrativní nastavení vlastností vazby

Nastavení vlastnosti vazby lze určit 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>
```

Jiné rozhraní než 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 pro určení režimu připojení klienta, musíte upravit kód služby tak, aby určoval vlastnost během inicializace vlastní vazby.

Chcete-li určit režim 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 194. Hodnoty vlastností vazby při administrativním nebo programovém nastavení

Název vlastnosti	Aplikace klienta nebo služby	Administrativní hodnota	Programová hodnota	Popis
maxBufferPoolSize	Oboje	0 až 64bitové celé číslo se znaménkem	0 až 64bitové celé číslo se znaménkem	Určuje maximální velikost paměti, kterou lze použít k uložení vyrovnávacích pamětí zpráv WCF pro instanci kanálu.
Velikost maxMessage	Oboje	1 až 32bitové celé číslo se znaménkem	1 až 32bitové celé číslo se znaménkem	Určuje maximální velikost paměti, kterou lze použít pro jednotlivé zprávy WCF.
Režim clientConnection	Oboje	0 (Výchozí hodnota) 1	AS_URI (výchozí hodnota) KLIENT_NESPRAVOVANÝ	Určuje režim připojení klienta transportního kanálu. Hodnota `0` znamená, že režim připojení klienta je uveden v identifikátoru URI. Používá se pouze v případě, že je použito připojení klienta. Určuje, že režim připojení klienta je uveden v identifikátoru URI. 0 je výchozí hodnota, pokud není nastaven žádný režim připojení klienta. 1 znamená, že režim připojení klienta je nespravovaný klient. Používá se pouze v případě, že je použito připojení klienta.
MaxConcurrentvolání	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 současně. Je-li spuštěno více operací, jsou řazeny do fronty, dokud neskončí probíhající operace nebo dokud neuplyne časový limit. Toto nastavení lze použít k řízení maximálního počtu podprocesů a prostředků, které může spotřebovat jednotlivý server proxy. `0` odebere toto omezení a povolí souběžné provádění všech operací.

Tabulka 194. Hodnoty vlastností vazby při administrativním nebo programovém nastavení (pokračování)				
Název vlastnosti	**Aplikace klienta nebo služby**	**Administrativní hodnota**	**Programová hodnota**	**Popis**
MaxConcurrentvolání	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 zajištěném doručení viz “WCF vlastní kanál zajištěné doručení” na stránce 1229). Určuje maximální počet souběžných operací, které mohou pro daný koncový bod současně probíhat. 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 pro akci požadavků. Nadměrná alokace může být kontraproduktivní a může mít závažný vliv na výkon. Pro podporu této vlastnosti musí být provedena odpovídající konfigurace fondu podprocesů.

Stavební a hostingové služby pro WCF

Přehled služeb Microsoft Windows Communication Foundation (WCF) vysvětlující, jak vytvářet a konfigurovat služby WCF.

Vlastní kanál IBM MQ pro služby WCF a služby WCF, které jej používají, může být hostován následujícími metodami:

- Vlastní hostování
- Windows Služba

Vlastní kanál IBM MQ pro WCF nemůže být hostován ve službě Windows Process Activation Service.

Následující témata poskytují několik jednoduchých příkladů samohostování, které demonstrují příslušné kroky. Online dokumentaci k 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: Samohostování administrativně pomocí konfiguračního souboru aplikace

Po vytvoření konfiguračního souboru aplikace otevřete instanci služby a přidejte uvedený kód do aplikace.

Než začnete

Vytvořte nebo upravte konfigurační soubor aplikace pro službu, jak je popsáno v tématu: “Vytvoření vlastního kanálu WCF administrativně poskytnutím informací o vazbách a koncových bodech v konfiguračním souboru aplikace” na stránce 1234

Informace o této úloze

1. Vytvořte instanci a otevřete instanci služby v hostiteli služby. Typ služby musí být stejný jako typ služby uvedený v konfiguračním souboru služby.
2. Do aplikace přidejte následující kód:

```
ServiceHost service = new ServiceHost(typeof(MyService));
service.Open();
...
service.Close();
```

Sestavení aplikací služeb WCF pomocí metody 2: Samohostování programově přímo z aplikace

Přidejte vlastnosti vazby, vytvořte hostitele služby s instancí požadované třídy služeb a otevřete službu.

Než začnete

1. Přidejte odkaz na soubor `IBM.XMS.WCF.dll` vlastního kanálu do projektu. Adresář `IBM.XMS.WCF.dll` se nachází v adresáři *`WMQInstallDir`*`\bin` , kde *WMQInstallDir* je adresář, ve kterém je nainstalován produkt IBM MQ .
2. Přidejte *pomocí příkazu* do oboru názvů `IBM.XMS.WCF` , například: `using IBM.XMS.WCF`
3. Vytvořte instanci prvku vazby kanálů a koncového bodu, jak je popsáno v tématu: “Vytvoření vlastního kanálu WCF programovým potlačením informací o vazbách a koncových bodech” na stránce 1236

Informace o této úloze

Pokud jsou vyžadovány změny vlastností vazby kanálu, postupujte takto:

1. Přidejte vlastnosti vazby do souboru `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 konfigurována pro použití vlastního kanálu IBM MQ pro WCF.

Informace o této úloze

Pokud musí být metadata služeb vystavena (aby k nim nástroje, jako např. `svcutil` , mohly přistupovat přímo ze spuštěné služby, a nikoli například z offline souboru WSDL), musí být vystavena metadata služeb pomocí koncového bodu HTTP . K přidání tohoto dalšího koncového bodu lze použít následující kroky.

1. Přidejte základní adresu, kde musí být metadata vystavena ServiceHost, například:

```
ServiceHost service = new ServiceHost(typeof(TestService),
                      new Uri("http://localhost:8000/MyService"));
```

2. Před otevřením služby přidejte do ServiceHost následující kód:

```
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 následující adrese: `http://localhost:8000/MyService`

Sestavení klientských aplikací pro WCF

Přehled generování a sestavení aplikací klienta Microsoft Windows Communication Foundation (WCF).

Klientská aplikace může být vytvořena pro službu WCF; klientské aplikace jsou obvykle generovány pomocí nástroje Microsoft ServiceModel Metadata Utility Tool (Svcutil.exe) k vytvoření požadovaných konfiguračních souborů a souborů proxy, které mohou být použity přímo aplikací.

Generování konfiguračních souborů serveru proxy klienta WCF a aplikace pomocí nástroje svcutil s metadaty ze spuštěné služby

Pokyny pro použití nástroje Microsoft svcutil.exe ke generování klienta pro službu, která je konfigurována pro použití vlastního kanálu IBM MQ pro WCF.

Než začnete

Existují tři nezbytné předpoklady pro použití nástroje svcutil k vytvoření požadovaných konfiguračních souborů a souborů proxy, které mohou být použity přímo aplikací:

- Služba WCF musí být spuštěna před spuštěním nástroje svcutil.
- Služba WCF musí vystavit svá metadata pomocí portu HTTP kromě odkazů na vlastní koncový bod kanálu IBM MQ , aby generovala 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 IBM MQ , ale také vystavuje metadata za běhu prostřednictvím odděleného portu HTTP :

1. Spusťte službu WCF (služba musí být spuštěna před spuštěním nástroje svcutil).
2. Přidejte podrobnosti z konfiguračního souboru `svcutil.exe` z kořenového adresáře instalace do aktivního konfiguračního souboru svcutil, obvykle `C:\Program Files\Microsoft SDKs\Windows\v6.0A\bin\svcutil.exe.config` , aby svcutil rozpoznal 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 příkaz svcutil ke generování klientských souborů z wsdl. Další informace viz: “Generování konfiguračních souborů serveru proxy klienta WCF a aplikace pomocí nástroje svcutil s WSDL” na stránce 1244

Generování konfiguračních souborů serveru proxy klienta WCF a aplikace pomocí nástroje svcutil s WSDL

Pokyny pro generování klientů WCF z WSDL, pokud nejsou metadata služby k dispozici.

Pokud metadata služby nelze přímo načíst pro generování klienta z metadat ze spuštěné služby, pak lze místo toho použít svcutil ke generování klientských souborů z WSDL. Ve WSDL je třeba provést následující úpravy, aby bylo možné určit, že má být použit vlastní kanál IBM MQ :

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 zásad, a odeberte všechny definice `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
```

Sestavení 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 kanálu WCF administrativně poskytnutím informací o vazbách a koncových bodech v konfiguračním souboru aplikace” na stránce 1234

Informace o této úloze

Vytvořte instanci a otevřete instanci serveru proxy klienta. Parametr předaný generovanému serveru proxy musí být stejný jako název koncového bodu uvedený 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();
            }
```

Sestavení aplikací klienta WCF pomocí serveru proxy klienta s programovou konfigurací

Než začnete

1. Přidejte odkaz na soubor `IBM.XMS.WCF.dll` vlastního kanálu do projektu. Soubor `IBM.XMS.WCF.dll` se nachází v adresáři *WMQInstallDir*`\bin` , kde *WMQInstallDir* je adresář, ve kterém je nainstalován produkt IBM MQ .
2. Přidejte *pomocí příkazu* do oboru názvů `IBM.XMS.WCF` , například: `using IBM.XMS.WCF`
3. Vytvořte instanci prvku vazby th ' a koncového bodu kanálu, jak je popsáno v tématu: “Vytvoření vlastního kanálu WCF programovým potlačením informací o vazbách a koncových bodech” na stránce 1236

Informace o této úloze

Jsou-li vyžadovány změny vlastností vazby kanálu, postupujte takto.

1. Přidejte vlastnosti vazby do souboru `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 server proxy klienta, jak ukazuje následující obrázek, kde *vazba* a *adresa koncového bodu* jsou vazba a adresa koncového bodu nakonfigurované v kroku 1 a předané:

```
 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 Windows Communication Foundation (WCF) poskytují několik jednoduchých příkladů, jak lze použít vlastní kanál IBM MQ .

Chcete-li sestavit ukázkové projekty, potřebujete buď sadu SDK Microsoft.NET 3.5 , nebo produkt Microsoft Visual Studio 2008.

Ukázka jednoduchého jednosměrného klienta a serveru WCF

Tato ukázka demonstruje vlastní kanál IBM MQ používaný ke spuštění služby WCF (Windows Communication Foundation) z klienta WCF s použitím jednosměrného tvaru kanálu.

Informace o této úloze

Služba implementuje jedinou metodu, jejímž výstupem je řetězec na konzolu. Klient byl vygenerován pomocí nástroje `svcutil` k načtení metadat služby ze samostatně vystaveného koncového bodu HTTP , jak je popsáno v tématu "Generování konfiguračních souborů serveru proxy klienta WCF a aplikace pomocí nástroje svcutil s metadaty ze spuštěné služby" na stránce 1243 .

Ukázka byla konfigurována se specifickými názvy prostředků, jak je popsáno v následujícím postupu. Pokud musíte 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 JMS viz *IBM MQ Přenos pro SOAP* v dokumentaci produktu IBM MQ . Potřebujete-li upravit ukázkové řešení a zdroj, potřebujete prostředí IDE, například Microsoft Visual Studio 8 nebo vyšší.

Postup

1. Vytvořte správce front s názvem *QM1*
2. Vytvořte cíl fronty s názvem *SampleQ*
3. Spusťte službu, 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 aplikace pětkrát odesílá pět zpráv do *SampleQ*

Výsledky

Aplikace služeb získá zprávy z *SampleQ* a zobrazí `Hello World` na obrazovce pětkrát.

Jak pokračovat dále

Ukázka klienta a serveru WCF s jednoduchou odpovědí na požadavek

Tato ukázka demonstruje vlastní kanál IBM MQ používaný ke spuštění služby WCF (Windows Communication Foundation) z klienta WCF pomocí tvaru kanálu požadavek-odezva.

Informace o této úloze

Tato služba poskytuje několik jednoduchých metod kalkulačky pro přidání a odečtení dvou čísel, a pak vrátit výsledek. Klient byl vygenerován pomocí nástroje `svcutil` k načtení metadat služby ze samostatně vystaveného koncového bodu HTTP , jak je popsáno v tématu "Generování konfiguračních souborů serveru proxy klienta WCF a aplikace pomocí nástroje svcutil s metadaty ze spuštěné služby" na stránce 1243 .

Ukázka byla konfigurována se specifickými názvy prostředků, jak je popsáno v následujícím postupu. Potřebujete-li změnit názvy prostředků, musíte také změnit odpovídající hodnotu v aplikaci klienta 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 JMS viz *IBM MQ Přenos pro SOAP* v dokumentaci produktu IBM MQ . Potřebujete-li upravit ukázkové řešení a zdroj, potřebujete prostředí IDE, například Microsoft Visual Studio 8 nebo vyšší.

Postup

1. Vytvořte správce front s názvem *QM1*
2. Vytvořte cíl fronty s názvem *SampleQ*
3. Vytvořte cíl fronty s názvem *SampleReplyQ*
4. Spusťte službu, 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

Po spuštění klienta je spuštěn následující proces a opakuje se čtyřikrát, takže je odesláno celkem pět zpráv:

1. Klient vloží zprávu požadavku do *SampleQ* a čeká na odpověď.
2. Služba získá zprávu požadavku z *SampleQ*.
3. Služba přidá a odečte některé hodnoty pomocí obsahu zprávy.
4. Služba poté vloží výsledky do zprávy v *SampleReplyQ*a čeká, až klient vloží novou zprávu.
5. Klient obdrží zprávu z *SampleReplyQ* a zobrazí výsledky na obrazovce.

Jak pokračovat dále

Ukázka klienta WCF pro službu .NET , jejímž hostitelem je IBM MQ

Ukázkové aplikace klienta a ukázkové aplikace serveru proxy služby jsou dodávány jak pro produkt .NET , tak pro produkt Java. Ukázky jsou založeny na službě Stock Quote, která přijímá žádost o cenovou nabídku a poté poskytuje cenovou nabídku.

Než začnete

Ukázka vyžaduje, aby bylo prostředí hostitele služby .NET SOAP over JMS správně nainstalováno a nakonfigurováno v produktu IBM MQ a bylo přístupné z lokálního správce front.

Je-li prostředí .NET SOAP přes JMS správně nainstalováno a nakonfigurováno v produktu IBM MQ a je-li přístupné z lokálního správce front, musí být provedeny další konfigurační kroky.

1. Nastavte proměnnou prostředí **WMQSOAP_HOME** na instalační adresář IBM MQ , například: `C:\Program Files\IBM\MQ`
2. Ujistěte se, že je Java kompilátor `javac` k dispozici a na cestě PATH.
3. Zkopírujte soubor `axis.jar` z adresáře `prereqs/axis` obrazu instalace do produkčního adresáře IBM MQ , například: `C:\Program Files\IBM\MQ\java\lib\soap`
4. Přidejte do 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 správně uvedeno v produktu *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 souboru .NET lze zadat například: `set msfwdir=%ProgramFiles%\Microsoft Visual Studio .NET 2003\SDK\v1.1\Bin`

Po dokončení předchozích kroků službu otestujte a spusťte:

1. Přejděte do svého pracovního adresáře SOAP přes JMS .
2. Chcete-li spustit ověřovací test a nechat modul listener služby spuštěný, zadejte jeden z následujících příkazů:
   - 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 ponechává listenery spuštěné po dokončení testu.

Pokud jsou během této konfigurace nahlášeny chyby, můžete odebrat všechny změny, aby mohla být procedura restartována následujícím způsobem:

1. Odstraňte vygenerovaný protokol SOAP přes adresář JMS .
2. Odstraňte správce front.

Informace o této úloze

Tato ukázka demonstruje připojení klienta WCF k ukázkové službě .NET SOAP přes JMS poskytované v produktu IBM MQ s použitím tvaru jednosměrného kanálu. Služba implementuje jednoduchý příklad StockQuote , jehož výstupem je textový řetězec na konzolu.

Klient byl vygenerován pomocí WSDL pro generování klientských souborů, jak je popsáno v tématu “Generování konfiguračních souborů serveru proxy klienta WCF a aplikace pomocí nástroje svcutil s WSDL” na stránce 1244 .

Ukázka byla konfigurována se specifickými názvy prostředků, jak je popsáno v následujícím postupu. Potřebujete-li změnit názvy prostředků, musíte také změnit odpovídající hodnotu v aplikaci klienta v souboru *MQ_INSTALLATION_PATH* \tools\wcf\samples\WMQNET\default\client\app.config a v aplikaci služeb 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 JMS viz *IBM MQ Přenos pro SOAP* v dokumentaci produktu IBM MQ .

Postup

Spusťte 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 aplikace pětkrát odesílá pět zpráv do ukázkové fronty.

Výsledky

Aplikace služeb získá zprávy z ukázkové fronty a zobrazí Hello World pětkrát na obrazovce.

Ukázka klienta WCF pro službu Axis Java , jejímž hostitelem je IBM MQ

Ukázkové aplikace klienta a ukázkové aplikace serveru proxy služby jsou dodávány jak pro produkt Java , tak pro produkt .NET. Ukázky jsou založeny na službě Stock Quote, která přijímá žádost o cenovou nabídku a poté poskytuje cenovou nabídku.

Než začnete

Tato ukázka vyžaduje, aby bylo prostředí hostitele služby .NET SOAP over JMS správně nainstalováno a nakonfigurováno v produktu IBM MQ a bylo přístupné z lokálního správce front.

Je-li prostředí .NET SOAP přes JMS správně nainstalováno a nakonfigurováno v produktu IBM MQ a je-li přístupné z lokálního správce front, musí být provedeny další konfigurační kroky.

1. Nastavte proměnnou prostředí **WMQSOAP_HOME** na instalační adresář IBM MQ , například: `C:\Program Files\IBM\MQ`
2. Ujistěte se, že je Java kompilátor `javac` k dispozici a na cestě PATH.
3. Zkopírujte soubor `axis.jar` z adresáře `prereqs/axis` obrazu instalace do instalačního adresáře IBM MQ .
4. Přidejte do 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 správně uvedeno v produktu *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 souboru .NET lze zadat například: `set msfwdir=%ProgramFiles%\Microsoft Visual Studio .NET 2003\SDK\v1.1\Bin`

Po dokončení předchozích kroků službu otestujte a spusťte:

1. Přejděte do svého pracovního adresáře SOAP přes JMS .
2. Chcete-li spustit ověřovací test a nechat modul listener služby spuštěný, zadejte jeden z následujících příkazů:
   - 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` ponechává listenery spuštěné po dokončení testu.

Pokud jsou během této konfigurace nahlášeny chyby, můžete odebrat všechny změny, aby se procedura restartovala následujícím způsobem:

1. Odstraňte vygenerovaný protokol SOAP přes adresář JMS .
2. Odstraňte správce front.

Informace o této úloze

Ukázka demonstruje připojení klienta WCF k ukázkové službě Axis Java SOAP over JMS poskytované v produktu IBM MQ pomocí tvaru jednosměrného kanálu. Služba implementuje jednoduchý příklad StockQuote , jehož výstupem je textový řetězec do souboru, který je uložen v aktuálním adresáři.

Klient byl vygenerován pomocí WSDL pro generování klientských souborů, jak je popsáno v tématu “Generování konfiguračních souborů serveru proxy klienta WCF a aplikace pomocí nástroje svcutil s WSDL” na stránce 1244 .

Ukázka byla konfigurována se specifickými názvy prostředků, jak je popsáno v tomto odstavci. Potřebujete-li změnit názvy prostředků, musíte také změnit odpovídající hodnotu v aplikaci klienta v souboru *MQ_INSTALLATION_PATH* `\tools\wcf\samples\WMQAxis\default\client\app.config` a v aplikaci služeb 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

Spusťte 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 aplikace pětkrát odesílá pět zpráv do ukázkové fronty.

Výsledky

Aplikace služby získá zprávy z ukázkové fronty a přidá soubor `Hello World` pětkrát do souboru v aktuálním adresáři.

Ukázka klienta WCF na službu Java hostovanou WebSphere Application Server

Ukázkové aplikace klienta a ukázkové aplikace serveru proxy služby jsou dodávány pro produkt WebSphere Application Server 6. K dispozici je také služba typu požadavek-odezva.

Než začnete

Tato ukázka vyžaduje použití následující konfigurace IBM MQ :

Tabulka 195. IBM MQ požadovaná konfigurace

Objekt	**Požadované jméno**
Správce front	QM1
Lokální fronta	HelloWorld
Lokální fronta	Odpověď HelloWorld

Tato ukázka také vyžaduje, aby bylo hostitelské prostředí WebSphere Application Server 6 správně nainstalováno a nakonfigurováno. Produkt WebSphere Application Server 6 standardně používá připojení v režimu vazeb pro připojení k produktu IBM MQ . Proto musí být produkt WebSphere Application Server 6 nainstalován na stejném počítači jako správce front.

Po konfiguraci prostředí WAS musí být dokončeny následující další kroky konfigurace:

1. Vytvořte následující objekty rozhraní JNDI v úložišti rozhraní JNDI WebSphere Application Server :
   a. Cíl fronty JMS s názvem `HelloWorld`
      - Nastavte název rozhraní JNDI na hodnotu `jms/HelloWorld`
      - Nastavit název fronty na `HelloWorld`
   b. JMS továrna připojení fronty s názvem `HelloWorldQCF`
      - Nastavte název rozhraní JNDI na hodnotu `jms/HelloWorldQCF`
      - Nastavit název správce front na `QM1`
   c. JMS továrna připojení fronty 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 souboru WebSphere Application Server s následující konfigurací:
   - Nastavte název rozhraní JNDI továrny připojení na `jms/HelloWorldQCF`
   - Nastavte název cílového rozhraní JNDI na hodnotu `jms/HelloWorld`
3. Nainstalujte aplikaci `HelloWorldEJBEAR.ear` webové služby na server WebSphere Application Server 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 službu otestujte tak, že ji jednou spustíte:

1. Přejděte do pracovního adresáře Soap over JMS .
2. Zadáním tohoto příkazu spusťte ukázku: *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 klienta WCF k ukázkové službě WebSphere Application Server SOAP over JMS poskytované v ukázkách WCF, které jsou součástí produktu IBM MQ, pomocí tvaru kanálu požadavek-odezva. Tok zpráv mezi WCF a WebSphere Application Server pomocí IBM MQ front. Služba implementuje metodu `HelloWorld(...)` , která vezme řetězec a vrátí pozdrav klientovi.

Klient byl vygenerován pomocí nástroje svcutil k načtení metadat služby ze samostatně vystaveného koncového bodu HTTP , jak je popsáno v tématu “Generování konfiguračních souborů serveru proxy klienta WCF a aplikace pomocí nástroje svcutil s metadaty ze spuštěné služby” na stránce 1243 .

Ukázka byla konfigurována se specifickými názvy prostředků, jak je popsáno v následujícím postupu. Potřebujete-li změnit názvy prostředků, musíte také změnit odpovídající hodnotu v aplikaci klienta v souboru *MQ_INSTALLATION_PATH* `\tools\wcf\samples\WAS\default\client\app.config` a v aplikaci služeb v *MQ_INSTALLATION_PATH* `\tools\wcf\samples\WAS\HelloWorldsEJBEAR.ear` , kde *MQ_INSTALLATION_PATH* je instalační adresář IBM MQ.

Služba a klient jsou založeny na službě a klientovi uvedeném v IBM Developer článku *Sestavení webové služby JMS pomocí SOAP přes JMS a WebSphere Studio*. Další informace o vývoji webových služeb SOAP prostřednictvím systému JMS , které jsou kompatibilní s vlastním kanálem IBM MQ WCF, naleznete v tématu 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.

Aplikace klienta spustí obě metody služby současně a odešle dvě zprávy do ukázkové fronty.

Výsledky

Aplikace služeb získává zprávy z ukázkové fronty a poskytuje odezvu na volání metody `HelloWorld(...)` , jehož výstupem je klientská aplikace do konzoly.

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 Spojeného království ani jiných zemí, ve kterých je takovéto vyjádření v rozporu s místními zákony: SPOLEČNOST INTERNATIONAL BUSINESS MACHINES CORPORATION TUTO PUBLIKACI POSKYTUJE "TAK, JAK 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. 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 interoperability softwaru, oddělení 49XA
3605 Dálnice 52 N

Rochester, MN 55901
U.S.A.

Poskytnutí takových informací může být podmíněno dodržením určitých podmínek a požadavků zahrnujících v některých případech uhrazení stanoveného poplatku.

Licencovaný program popsaný v těchto informacích a veškerý licencovaný materiál, který je pro něj k dispozici, jsou poskytovány společností IBM na základě podmínek IBM Smlouvy se zákazníkem, IBM Mezinárodní licenční smlouvy pro programy 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 o programovacím rozhraní, jsou-li poskytnuty, jsou určeny k tomu, aby vám pomohly vytvořit aplikační software pro použití s tímto programem.

Tato příručka obsahuje informace o zamýšlených programovacích rozhraních, která zákazníkům umožňují 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: Tyto informace o diagnostice, úpravách a ladění nepoužívejte jako programovací rozhraní, protože se mohou měnit.

Ochranné známky

IBM, logo IBM , ibm.com, jsou ochranné známky společnosti IBM Corporation, registrované v mnoha jurisdikcích po celém světě. Aktuální seznam ochranných známek společnosti IBM je k dispozici na webu "Copyright and trademark information"www.ibm.com/legal/copytrade.shtml. Další 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 zahrnuje software vyvinutý projektem Eclipse (https://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: