7.5

Vývoj aplikací pro IBM WebSphere MQ

Poznámka

Než začnete používat tyto informace a produkt, který podporují, přečtěte si informace, které uvádí “Poznámky” na stránce 1089.

Toto vydání se vztahuje k verzi 7, vydání 5 produktu IBM® WebSphere MQ a ke všem následujícím vydáním a modifikacím, dokud nebude v nových vydáních uvedeno jinak.

Obsah

Vývoj aplikací

Produkt IBM WebSphere MQ nabízí několik způsobů, jak vyvíjet aplikace k odesílání a přijímání zpráv, které potřebujete k podpoře vašich obchodních procesů. Také můžete vyvíjet aplikace pro správu správců front a souvisejících prostředků.

Než vyvinete aplikace pro produkt IBM WebSphere MQ, ujistěte se, že jste obeznámeni s koncepcemi v příručce IBM WebSphere MQ Technical overviewIBM WebSphere MQ Technical overview.

Můžete vyvíjet aplikace pro IBM WebSphere MQ v řadě různých programovacích jazyků. Informace o podporovaných programovacích jazycích a jejich funkcích naleznete v části “Rozhodování o tom, jaký programovací jazyk použít” na stránce 75.

Typy aplikací, které můžete pro produkt IBM WebSphere MQ napsat na různých platformách, najdete v níže uvedených sekcích.

Typy aplikací, které můžete zapsat pro produkt IBM WebSphere MQ

Tyto informace jsou o typech aplikací, které lze zapsat na IBM WebSphere MQ.

Produkty IBM WebSphere MQ jsou správci front a aktivátory aplikací. Podporují rozhraní IBM Message Queue Interface (MQI), prostřednictvím které programy mohou vkládat zprávy do fronty a získávat zprávy z fronty.

S produktem IBM WebSphere MQ pro jiné platformy nežz/OS můžete psát aplikace, které:

- Odešlete zprávy do jiných aplikací spuštěných pod stejným operačním systémem. Aplikace mohou být buď ve stejném nebo v jiném systému.
- Odesílat zprávy aplikacím, které jsou spuštěny na jiných platformách produktu IBM WebSphere MQ .
- Použití front zpráv ze systému CICS pro TXSeries pro systémy AIX, TXSeries pro systémy HP-UX, TXSeries pro Solaris a TXSeries pro systémy Windows .
- Použijte řazení zpráv do front z prostředí Encina pro systémy AIX, HP-UX, Solaris a Windows .
- Zařazení do fronty zpráv z produktu Tuxedo pro systémy AIX, AT & T, HP-UX, Solaris a Windows .
- Použijte produkt IBM WebSphere MQ jako správce transakcí koordinující aktualizace provedené externími správci prostředků v rámci jednotek práce produktu IBM WebSphere MQ . Následující externí správci prostředků jsou podporováni a jsou v souladu s rozhraním XA sdružení X/OPEN
  - DB2
  - Informix
  - Oracle
  - Sybase
- Zpracovat několik zpráv dohromady jako jedinou jednotku práce, kterou lze potvrdit nebo zazálohovat.
- Spusťte z úplného prostředí produktu IBM WebSphere MQ nebo jej spusťte z prostředí klienta IBM WebSphere MQ MQI na následujících platformách:
  - UNIX and Linux® systémy
  - Windows

Související pojmy

Zabezpečení

Koncepty vývoje aplikací

K zápisu aplikací IBM WebSphere MQ můžete použít volbu procedurálních nebo objektově orientovaných jazyků. Informace o konceptech produktu IBM WebSphere MQ , které jsou užitečné pro vývojáře aplikací, naleznete v odkazech v tomto tématu.

Před zahájením návrhu a zápisu aplikací produktu IBM WebSphere MQ se seznamte se základními koncepcemi produktu IBM WebSphere MQ , viz témata v tématu Technický přehled. Informace o typech aplikací, které můžete psát pro produkt IBM WebSphere MQ, naleznete v příručce “Vývoj aplikací” na stránce 7.

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

- “Zprávy produktu IBM WebSphere MQ” na stránce 9
- Dvoubodový systém zpráv
- Úvod do systému zpráv publikování a odběru produktu WebSphere MQ
- “Použití rozhraní MQI (Message Queue Interface) v klientské aplikaci” na stránce 338
- “Použití webových služeb v produktu WebSphere MQ” na stránce 913
- “Kanály-uživatelské programy pro kanály systému zpráv” na stránce 380
- “Scénáře transakčního podpory” na stránce 39

Před spuštěním aplikací, které používají rozhraní MQI, je třeba vytvořit určité objekty produktu IBM WebSphere MQ . Další informace viz “Aplikační programy používající rozhraní MQI” na stránce 9.

Související pojmy

“Návrh aplikací produktu IBM WebSphere MQ” na stránce 86
Když se rozhodnete, jak aplikace mohou využívat výhody platforem a prostředí, které máte k dispozici, musíte se rozhodnout, jak budete používat funkce nabízené produktem WebSphere MQ.

“Ukázka programů WebSphere MQ” na stránce 92
Tato kolekce témat slouží k získání informací o ukázkových programech WebSphere MQ na různých platformách.

“Psaní front aplikace” na stránce 187
Prostřednictvím těchto informací můžete získat informace o vytváření aplikací pro řazení do front, připojování a odpojování od správce front, publikování a odběru a otevírání a zavírání objektů.

“Tvorba klientských aplikací” na stránce 337
Co potřebujete vědět, chcete-li zapisovat klientské aplikace do produktu WebSphere MQ.

“Rozhodování o tom, jaký programovací jazyk použít” na stránce 75
Tyto informace použijte k vyhledání informací o programovacích jazycích a rámcích podporovaných produktem IBM WebSphere MQa o některých aspektech jejich použití.

“Použití tříd produktu WebSphere MQ pro službu JMS” na stránce 692
Třídy WebSphere MQ pro platformy JMS (WebSphere MQ Classes for JMS) jsou poskytovatelem rozhraní JMS dodávaným s produktem WebSphere MQ. Kromě implementace rozhraní definovaných v balíku javax.jms produkt WebSphere MQ Classes for JMS poskytuje dvě sady rozšíření rozhraní JMS API.

“Použití rozhraní modelu objektu Component Interface (WebSphere MQ Automation Classes for ActiveX)” na stránce 1000
Produkt WebSphere MQ Automation Classes for ActiveX (MQAX) jsou komponenty ActiveX , které poskytují třídy, které můžete použít ve své aplikaci k přístupu k produktu WebSphere MQ.

“Použití tříd produktu WebSphere MQ pro prostředí Java” na stránce 631
Třídy WebSphere MQ pro prostředí Java umožňují používat produkt WebSphere MQ v prostředí Java. A Java application can use either WebSphere MQ classes for Java or WebSphere MQ classes for JMS to access WebSphere MQ resources.

“Použití .NET” na stránce 540
Třídy WebSphere MQ pro prostředí .NET umožňují programu napsaným v programovacím rámci .NET pro připojení k produktu WebSphere MQ jako klienta WebSphere MQ MQI nebo k přímému připojení k serveru WebSphere MQ .

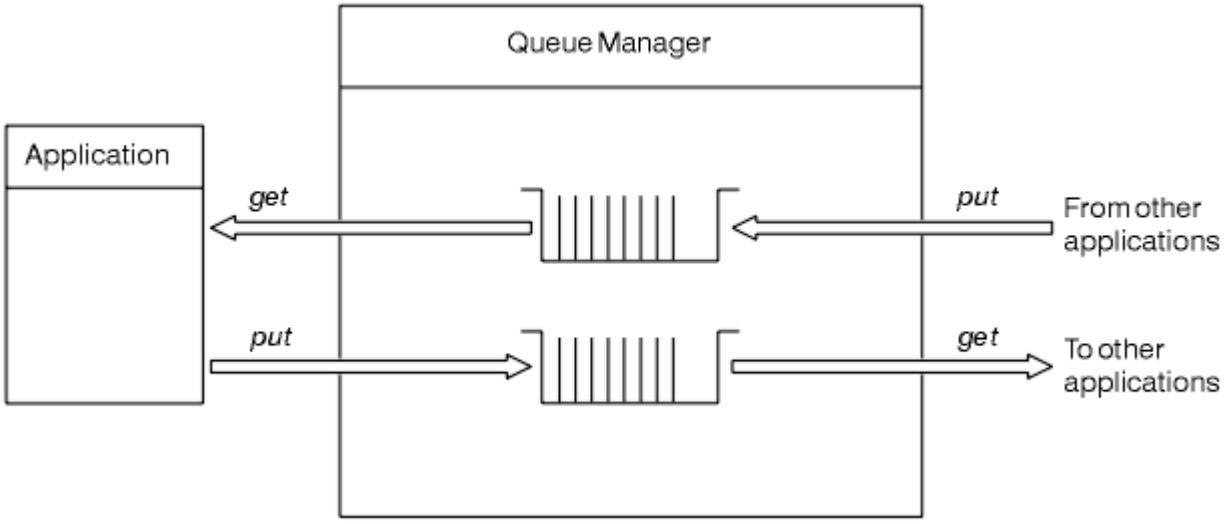
“Použití C++” na stránce 607
Produkt WebSphere MQ poskytuje třídy C + + ekvivalentní objekty WebSphere MQ a některé další třídy ekvivalentní k datovým typům pole. Poskytuje řadu funkcí, které nejsou prostřednictvím rozhraní MQI k dispozici.

“Sestavení aplikace IBM WebSphere MQ” na stránce 412
Tyto informace použijte k seznámení se s aplikací produktu IBM WebSphere MQ na různých platformách.

Aplikační programy používající rozhraní MQI

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

Produkt Obrázek 1 na stránce 9 zobrazuje aplikaci, která odebírá zprávy z fronty, zpracuje je a poté odešle některé výsledky do jiné fronty ve stejném správci front.



Obrázek 1. Fronty, zprávy a aplikace

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

Před spuštěním této aplikace musí být splněny následující podmínky:

- Správce front musí existovat a musí být spuštěn.
- Musí být definována první aplikační fronta, ze které se zprávy mají odstranit.
- Musí být definována také druhá fronta, na které aplikace vkládá zprávy.
- Aplikace musí být schopna připojit se ke správci front. Chcete-li tak učinit, musí být propojena s IBM WebSphere MQ. Viz “Sestavení aplikace IBM WebSphere MQ” na stránce 412.
- Aplikace, které vložila zprávy do první fronty, se musí také připojit ke správci front. Jsou-li vzdálené, musí být také nastaveny pomocí přenosových front a kanálů. Tato část systému není zobrazena v Obrázek 1 na stránce 9.

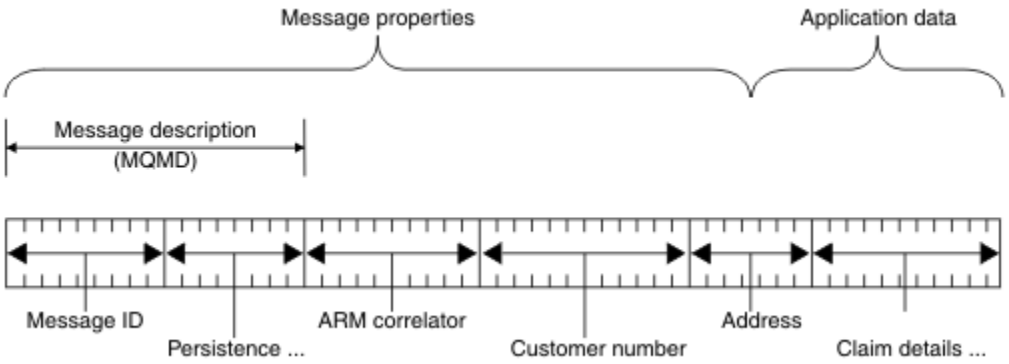
Zprávy produktu IBM WebSphere MQ

Tyto informace představují koncepci zpráv produktu IBM WebSphere MQ , části zpráv a deskriptor zpráv.

Zprávy IBM WebSphere MQ se skládají ze dvou částí:

- Vlastnosti zprávy
- Data aplikace

Obrázek 2 na stránce 10 představuje zprávu a ukazuje, jak je logicky rozdělena na vlastnosti zprávy a aplikační data.



Obrázek 2. Zastupování zprávy

Data aplikace přenášené ve zprávě produktu WebSphere MQ nejsou správcem front změněny, pokud se na něm neprovedou převod dat. Produkt WebSphere MQ kromě toho nevloží žádná omezení týkající se obsahu těchto dat. Délka dat v každé zprávě nesmí překročit hodnotu atributu *MaxMsgLength* jak ve frontě, tak i ve správci front.

V WebSphere MQ pro AIX, WebSphere MQ pro HP-UX, WebSphere MQ pro Linux, WebSphere MQ pro Solaris, a WebSphere MQ pro Okna, *MaxMsgLength* standardně zobrazuje 100 MB (104 857 600 bajtů).

V některých případech můžete zprávy mírně zkrátit, než je hodnota atributu *MaxMsgLength* . Další informace viz “Data ve zprávě” na stránce 221.

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

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

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

Popis způsobu použití polí v rámci MQMD, který obsahuje informace o původu zprávy, viz “kontext zprávy” na stránce 37 .

Jsou zde různé verze deskriptoru zpráv. Další informace o seskupování a segmentování zpráv (viz “Skupiny zpráv” na stránce 34) jsou poskytovány ve verzi 2 deskriptoru zpráv (nebo MQMDE). To je stejné jako deskriptor zprávy verze 1, ale obsahuje další pole. Ty jsou popsány v MQMMDE-rozšíření deskriptoru zpráv.

Typy zpráv

Produkt IBM WebSphere MQdefinuje čtyři typy zpráv.

Tyto čtyři zprávy jsou:

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

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

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

Datagramy

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

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

Zprávy požadavků

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

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

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

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

Odpovědi na zprávy

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

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

Hlášení zpráv

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

Mohou být generovány pomocí:

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

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

Typy zpráv sestav

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

- *Zpráva o výjimce*. Toto je odesláno jako odpověď na zprávu s nastaveným příznakem výjimek. Je generován agentem kanálu zpráv (MCA) nebo aplikací.

- *Zpráva o vypršení platnosti zprávy*. To označuje, že se aplikace pokusila načíst zprávu, která dosáhla své prahové hodnoty vypršení platnosti; zpráva je označena jako vyřazená. Tento typ sestavy je generován správcem front.
- *Potvrzení zprávy o příjmu (COA)*. Tato zpráva informuje o tom, že zpráva dosáhla cílové fronty. Je generován správcem front.
- *Potvrzení o potvrzení doručení (COD)*. To znamená, že zpráva byla načtena přijímající aplikací. Je generován správcem front.
- *Zpráva sestavy s kladným oznámením akce (PAN)*. To znamená, že požadavek byl úspěšně obsloužen (to znamená, že akce požadovaná ve zprávě byla úspěšně provedena). Tento typ sestavy je generován aplikací.
- *Zpráva sestavy Negative action notification (NAN)*. To znamená, že požadavek nebyl úspěšně obsloužen (to znamená, že akce požadovaná ve zprávě nebyla úspěšně provedena). Tento typ sestavy je generován aplikací.

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

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

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

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

Volby zprávy sestavy

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

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

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

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

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

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

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

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

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

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

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

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

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

Sestavy a segmentované zprávy

Nepodporováno na produktu WebSphere MQ pro z/OS .

Je-li zpráva segmentovaná (viz “Segmentace zpráv” na stránce 252 pro popis segmentovaných zpráv) a požádáte o generování sestav, můžete obdržet více sestav, než byste měli, kdyby zpráva nebyla segmentována.

Pro sestavy generované produktem WebSphere MQ

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

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

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

Pro sestavy generované aplikacemi

Pokud vaše aplikace generuje sestavy, vždy zkopírujte záhlaví WebSphere MQ , která se nacházejí na začátku původních dat zprávy, do dat zprávy sestavy.

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

Můžete rozpoznat záhlaví produktu WebSphere MQ , která musí být zkopírovaná, při pohledu na následující názvy formátu, počínaje MQMD a pokračujícím prostřednictvím všech přítomných záhlaví. Následující názvy `Format` označují tyto záhlaví WebSphere MQ :

- MQMDE

- MQDLH
- MQXQH
- MQIIH.
- MQH *

MQH* označuje libovolný název, který začíná znaky MQH.

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

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

Načítání sestav

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

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

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

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

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

Jeden přístup je následující:

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

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

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

Správci front nižší úrovně

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

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

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

Formát dat řízení zpráv a dat zprávy

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

Formát informací o řízení zpráv

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

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

Formát dat zprávy

Můžete určit kteroukoli z následujících možností:

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

Chcete-li to provést, použijte tato pole:

Format
Toto indikuje přijímači zprávy formát dat aplikace ve zprávě.

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

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

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

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

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

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

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

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

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

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

Poznámka: Zprávy s uvedeným parametrem MQFMT_NONE nejsou převedeny.

Převod v odesílajícím správci front

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

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

Formáty pro sestavení

Patří k nim:

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

  Produkt WebSphere MQ používá ve zprávách a událostech administrace Programovatelné zprávy formátu příkazů (v tomto případě se používá název formátu MQFMT_ADMIN). Pro své vlastní zprávy můžete použít stejný formát (s použitím názvu formátu MQFMT_PCF) a využívat výhod vestavěného převodu dat.

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

Formáty definované aplikací

Pro uživatelem definované formáty musí být převod dat aplikací prováděn ukončovacím programem konverze dat (další informace viz “Zápis uživatelských procedur pro převod dat” na stránce 399). V prostředí typu klient-server se uživatelská procedura načte na server a provede se převod.

Převod v přijímajícím správci front

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

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

Kódové znakové sady

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

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

U obecného převodu dat platí, že pokud operační systém podporuje kódové stránky DBCS, může jej produkt WebSphere MQ použít.

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

Při zápisu aplikací, které zasahují do více platforem, je třeba brát v úvahu převod dat aplikací, názvy formátů a uživatelské procedury. Informace o vyvolání a zápisu uživatelských procedur pro převod dat naleznete v příručce “Zápis uživatelských procedur pro převod dat” na stránce 399 .

Priority zpráv

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

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

Atribut *DefPriority* fronty nastavuje výchozí hodnotu priority pro zprávy vkládané do této fronty. Tato hodnota je nastavena při vytvoření fronty, ale lze ji později změnit. Alias front a lokální definice vzdálených front, mohou mít odlišné výchozí priority ze základních front, na které se tyto názvy řeší. Je-li v cestě rozpoznání více než jedna definice fronty (viz “Rozpoznání názvu” na stránce 208), je výchozí priorita převzata z hodnoty atributu *DefPriority* fronty zadané v otevřeném příkazu (v době operace vložení).

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

Vlastnosti zprávy

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

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

Použití vlastností zpráv v produktu WebSphere MQ imituje použití vlastností v rozhraní JMS. To znamená, že můžete nastavit vlastnosti v aplikaci platformy JMS a načíst je v procedurálním aplikaci WebSphere MQ nebo v opačném směru. Chcete-li zpřístupnit vlastnost pro aplikaci JMS, přiřaďte ji předponou "usr"; poté je k dispozici (bez předpony) jako vlastnost uživatele zprávy JMS. Například vlastnost WebSphere MQ *usr.myproperty* (znakový řetězec) je přístupná pro aplikaci JMS pomocí volání JMS `message.getStringProperty('myproperty')`. Všimněte si, že aplikace JMS nemohou přistoupit k vlastnostem s předponou "usr", pokud obsahují dva nebo více U+002E (".") znaků. Vlastnost bez předpony a ne U+002E (".") jsou považovány za předponu "usr", jako by se s předponou "usr". A naopak,

vlastnost uživatele nastavená v aplikaci JMS může být přístupná v aplikaci WebSphere MQ přidáním "usr". zadejte předponu názvu vlastnosti, která se bude provádět při volání MQINQMP.

Vlastnosti zprávy a délka zprávy

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

Obecně platí, že když pomocí příkazu MQSETMP nastavíte vlastnosti, velikost vlastnosti je délka názvu vlastnosti v bajtech, plus délka hodnoty vlastnosti v bajtech, jak byla předána do volání funkce MQSETMP. Je možné, že znaková sada názvu vlastnosti a hodnota vlastnosti se změní během přenosu zprávy na místo určení, protože je lze převést na Unicode; v tomto případě by se velikost vlastnosti mohla změnit.

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

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

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

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

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

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

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

Při toku do správce front, který je starší než verze 7.0 produktu, se vlastnosti zprávy, kromě těch v deskriptoru zpráv, započítávají do délky zprávy. Proto byste měli buď zvýšit hodnotu atributu *MaxMsgLength* kanálů, které se budou posílat do systému dříve, než je verze 7.0 , aby bylo možné kompenzovat skutečnost, že pro každou zprávu může být odesláno více dat. Případně můžete snížit délku fronty nebo správce front *MaxMsgLength*, aby byla celková úroveň dat posílaná po celém systému stejná.

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

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

Názvy vlastností

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

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

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

Vzhledem k tomu, že v rozhraní JMS neexistuje maximální délka názvu vlastnosti, je možné, aby aplikace JMS nastavila platný název vlastnosti JMS, který není platným názvem vlastnosti produktu WebSphere MQ , je-li uložen ve struktuře MQRFH2 .

V tomto případě jsou při analýze použity pouze prvních 4095 znaků názvu vlastnosti; následující znaky jsou ořízнuty. To by mohlo způsobit, že aplikace používá selektory, aby se shodovala s výběrovým řetězcem, nebo aby odpovídala řetězci, když se neočekává, protože více než jedna vlastnost by mohla ořízнout na stejný název. Je-li název vlastnosti zkrácen, produkt WebSphereMQ vydá zprávu s protokolem o chybě.

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

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

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

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

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

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

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

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

Na názvy vlastností se vztahují následující omezení:

1. Vlastnost nesmí začínat následujícími řetězci:
   - "JMS"-vyhrazeno pro použití třídami produktu WebSphere MQ pro platformu JMS.
   - "usr.JMS"-není platné.

   Jediné výjimky jsou následující vlastnosti poskytující synonyma pro vlastnosti JMS:

Vlastnost	**Synonymum**
JMSCorrelationID	Kořen .MQMD.CorrelId nebo jms.Cid
JMSDeliveryMode	Kořen .MQMD.Persistence nebo jms.Dlv

Vlastnost	Synonymum
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 Kořen .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 aplikacím MQI přistupovat k vlastnostem platformy JMS podobným způsobem jako třídy WebSphere MQ pro klientskou aplikaci JMS. Z těchto vlastností lze pomocí rozhraní MQI nastavit pouze JMSCorrelationID, JMSReplyTo, JMSTType, JMSXGroupIDa JMSXGroupSeq .

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

2. Vlastnost nesmí být volána ani v žádné směsi malých nebo velkých, "NULL", "TRUE", "FALSE", "NOT", "AND", "OR", "BETWEEN", "LIKE", "IN", "IS" a "ESCAPE". Tyto jsou názvy klíčových slov SQL použité ve výběrových řetězcích.
3. Název vlastnosti začínající "mq " v libovolné kombinaci malých a velkých písmen a ne začínající "mq_usr" může obsahovat pouze jeden "." character (U+002E). Vícenásobné "." znaky nejsou ve vlastnostech s těmito předponami povoleny.
4. Dva "." Znaky musí obsahovat jiné znaky mezi; nemůžete mít prázdný bod v hierarchii. Podobně název vlastnosti nesmí končit znakem "." Znak.
5. Pokud aplikace nastaví vlastnost "a.b" a pak vlastnost "a.b.c", není jasné, zda v hierarchii "b" obsahuje hodnotu nebo jinou logickou skupinu. Such a hierarchy is "mixed content" and this is not supported. Nastavení vlastnosti, která způsobí smíšený obsah, není povoleno.

Tato omezení jsou vynucována mechanismem ověřování následujícím způsobem:

- Názvy vlastností jsou ověřovány při nastavení vlastnosti pomocí volání funkce MQSETMP – nastavení vlastnosti zprávy , pokud bylo při vytvoření manipulátoru zprávy vyžadováno ověření platnosti. Pokud pokus o ověření platnosti vlastnosti je proveden a selže kvůli chybě ve specifikaci názvu vlastnosti, kód dokončení je MQCC_FAILED s důvodem:
  - Objekt MQRC_PROPERTY_NAME_ERROR z důvodů 1-4.
  - Objekt MQRC_MIXED_CONTENT_NOT_ALLOWED z důvodu 5.
- Názvy vlastností, které jsou zadány přímo jako prvky MQRFH2 , nejsou garantovány, aby byly ověřeny voláním MQPUT.

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

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

Název vlastnosti	Pole deskriptoru zpráv
Root.MQMD. < pole>	< Pole >

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

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

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

Pokud data zprávy začínají řetězcem MQMDE, který je uznán správcem front, lze k polím MQMDE přistupovat s použitím popsané notace `Root.MQMD.<Field>` . V tomto případě jsou pole MQMDE považována za logickou část MQMD z perspektivy vlastností. Viz část "MQMDE uvedená v MQPUT a MQPUT1 volání" v Přehled MQMDE.

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

Datový typ hodnoty vlastnosti musí být jedna z následujících hodnot:

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

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

Řetězec bajtů není platným datovým typem vlastnosti v rozhraní JMS nebo XMS. Doporučuje se nepoužívat vlastnosti bajtových řetězců ve složce < usr>.

Výběr zpráv z front

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

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

Výběr pomocí funkčních volání MQSUB a MQOPEN

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

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

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

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

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

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

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

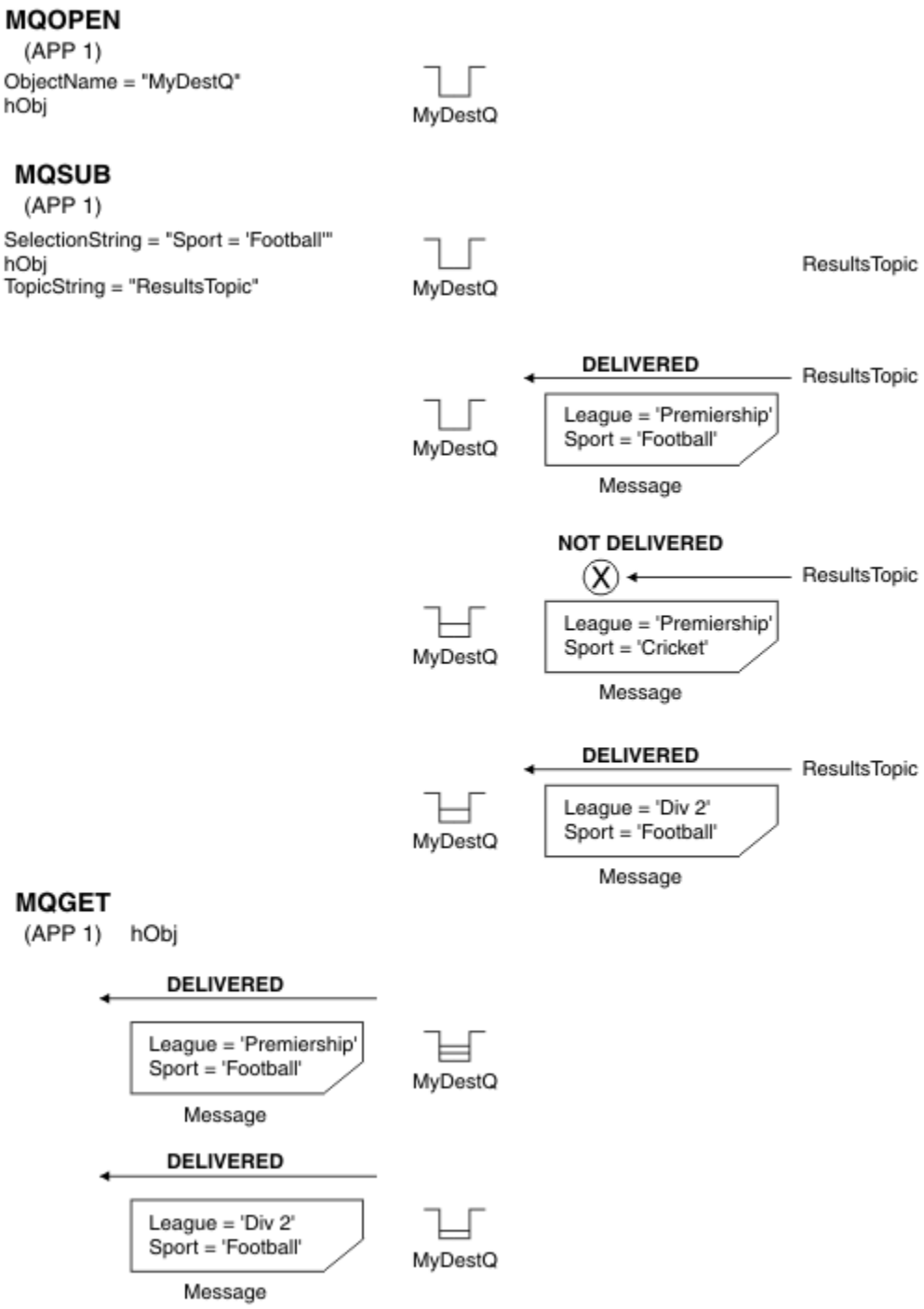
Hodnota SelectionString je vrácena ve výstupu z volání MQSUB s použitím volby MQSO_RESUME, pokud jste poskytli vyrovnávací paměť a v parametru VSBufSizeje kladná délka vyrovnávací paměti. Pokud vyrovnávací paměť nezadáte, vrátí se v poli VSLLength pole MQCHARV pouze délka řetězce výběru. Je-li poskytnutá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

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

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

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

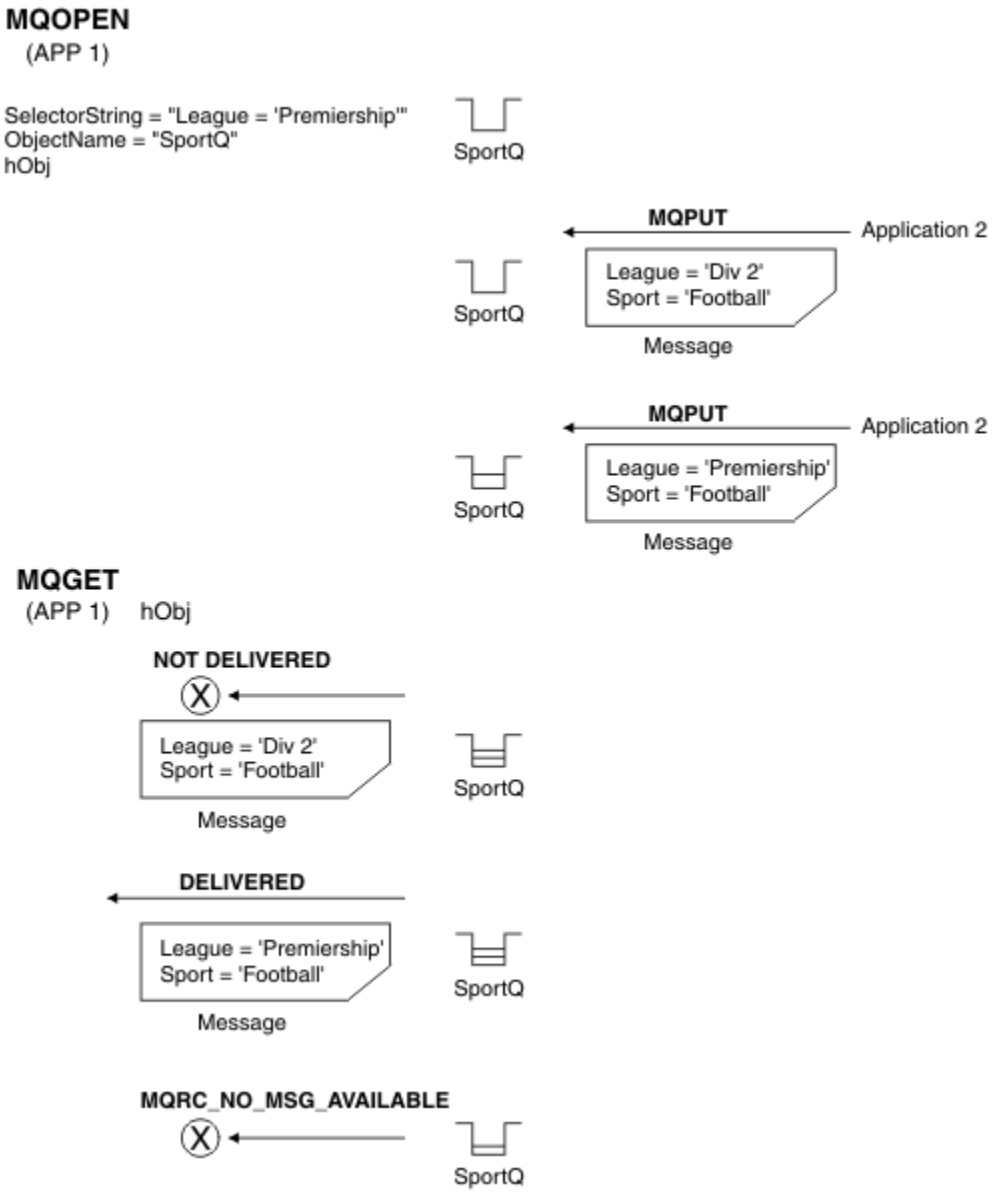
Produkt Obrázek 3 na stránce 23 zobrazuje proces výběru pomocí volání MQSUB.



Obrázek 3. Výběr pomocí volání MQSUB

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

Produkt Obrázek 4 na stránce 24 zobrazuje proces výběru pomocí volání MQOPEN.



Obrázek 4. Výběr pomocí volání MQOPEN

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

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

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

Chování výběru
Přehled chování výběru produktu IBM WebSphere MQ .

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

- Má formát MQFMT_MD_EXTENSION

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

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

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

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

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

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

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

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

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

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

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

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

Výkon systému zpráv

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

Chcete-li optimalizovat výběr zpráv v hlubokých frontách, je-li výběr založen na JMSCorrelationID nebo JMSMessageID, použijte výběrový řetězec ve tvaru `JMSCorrelationID = ...` nebo `JMSMessageID = ...` a odkazujte pouze na jednu vlastnost.

Tato metoda nabízí výrazné zlepšení výkonu pro výběr v JMSCorrelationID a nabízí mezní zlepšení výkonu pro JMSMessageID.

Použití komplexních selektorů

Selektory mohou obsahovat mnoho komponent, například:

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

Použití takových komplexních selektorů může mít vážný dopad na výkon a nadměrné požadavky na prostředky. Produkt IBM WebSphere MQ tak bude chránit systém tím, že nebude zpracovávat příliš složité selektory, které by mohly vést k nedostatku systémových prostředků. Ochrana se může vyskytnout po přibližně 100 testech na některých platformách, takže selektory blížící se k počtu komponent mohou vidět selhání. Doporučuje se, aby použití selektorů s mnoha komponentami bylo důkladně vyzkoušeno a otestováno na příslušných platformách, aby se zajistilo, že nebudou dosaženy limity ochrany.

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

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

Související pojmy

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

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

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

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

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

Selektor může obsahovat:

- Literály:
  - Řetězcové literály jsou uzavřeny v jednoduchých uvozovkách. Dvě po sobě jdoucí jednoduché uvozovky představují jednoduché uvozovky. Příklady jsou 'literal' a 'literal''s '. Podobně jako řetězcové literály Java tyto používají kódování znaků Unicode. Dvojité uvozovky nelze použít k uzavření řetězcového literálu. Libovolnou posloupnost bajtů lze použít mezi jednoduchými uvozovkami.
  - Bajtový řetězec je jeden nebo více dvojic hexadecimálních znaků uzavřených do uvozovek a s předponou 0x. Příklady jsou "0x2F1C" nebo "0XD43A". Délka bajtového řetězce musí být alespoň jeden bajt. Pokud je řetězec bajtů selektoru porovnán se vlastností zprávy typu MQTYPE_BYTE_STRING, neprovedou se žádné speciální akce na úvodní nebo koncové nule. S bajty se zachází jako s jiným znakem. Endianness se také nezvažuje. Délka obou bajtových řetězců selektoru i vlastností musí být stejná, a posloupnost bajtů musí být stejná.

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

    - "myBytes = "0x0AFC23"" = TRUE

    Následující výběry řetězců se neshodují:

    - "myBytes = "0xAFC23"" = MQRC_SELECTOR_SYNTAX_ERROR (protože počet bajtů není násobkem dvou)
    - "myBytes = "0x0AFC2300"" = FALSE (protože koncová nula je významná v porovnání)

- "myBytes = "0x000AFC23"" = FALSE (protože vedoucí nula je významná v porovnání)
- "myBytes = "0x23FC0A"" = FALSE (protože endianness se neuvažuje)

– Hexadecimální čísla začínají nulou a následují velkými nebo malými písmeny x. Zbytek literálu obsahuje jeden nebo více platných hexadecimálních znaků. Příklady: 0xA, 0xAF, 0X2020.
– Počáteční nula následovaná jednou nebo více číslicemi v rozsahu 0-7 je vždy interpretována jako začátek oktalového čísla. Nemůže představovat desítkové číslo s předponou nula, například 09 , vrací chybu syntaxe, protože číslo 9 není platnou oktalovou číslicí. Příklady oktalových čísel jsou 0177, 0713.
– Přesný číselný literál je číselná hodnota bez desetinné čárky, jako například 57, -957a +62. Přesný číselný literál může mít koncový znak L nebo malá písmena L; to nemá vliv na to, jak je číslo uloženo nebo interpretováno. Produkt WebSphere MQ podporuje přesné číslice v rozsahu -9,223,372,036,854,775,808 až 9,223,372,036,854,775,807.
– Přibližný numerický literál je číselná hodnota ve vědecké notaci, jako například 7E3 nebo -57.9E2, nebo číselná hodnota s desítkovým číslem, například 7., -95.7nebo +6.2. Produkt WebSphere MQ podporuje čísla v rozsahu -1.797693134862315E+308 až 1.797693134862315E+308.

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

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

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

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

- Identifikátory:

  Identifikátor je posloupnost znaků s proměnnou délkou, která musí začínat platným znakem začátku identifikátoru následován nulou nebo více platnými znaky identifikátoru. Pravidla pro názvy identifikátorů jsou stejná jako pravidla pro názvy vlastností zpráv, viz “Názvy vlastností” na stránce 18 a “Omezení názvu vlastnosti” na stránce 19 , kde získáte další informace.

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

  Identifikátory jsou buď odkazy na pole záhlaví, nebo odkazy na vlastnosti. Typ hodnoty vlastnosti v selektoru zpráv musí odpovídat typu použitému k nastavení vlastnosti, ačkoli je to možné, je-li to možné, je provedena číselná povýšení. Dojde-li k neshodě typů, pak je výsledkem výrazu FALSE. Pokud na vlastnost, která ve zprávě neexistuje, se odkazuje, její hodnota je NULL.

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

  Názvy polí JMS a názvy vlastností, které jsou mapovány na názvy vlastností nebo názvy polí MQMD , jsou platnými identifikátory v řetězci výběru. Produkt WebSphere MQ mapuje rozpoznávané pole JMS a názvy vlastností na hodnoty vlastností zprávy. Další informace viz “Selektory zpráv v JMS” na stránce 781. Jako příklad, řetězec výběru "JMSPriority >=" se vybere ve vlastnosti Pri ve složce jms aktuální zprávy.

- Přetečení/podtečení:

  Pro desetinné i přibližné číselné čísla nejsou definovány následující:

  – Určení čísla, které je mimo definovaný rozsah

  - Určení aritmetického výrazu, který by způsobil přetečení nebo podtečení

  Pro tyto podmínky nejsou provedeny žádné kontroly.
- Netisknutelné:

  Definováno jako mezera, posun o znak, nový řádek, návrat vozíku, vodorovný tabelátor nebo vertikální tabulátor. Následující znaky Unicode jsou rozpoznány jako bílé znaky:
  - `\u0009 to \u000D`
  - `\u0020`
  - `\u001C`
  - `\u001D`
  - `\u001E`
  - `\u001F`
  - `\u1680`
  - `\u180E`
  - `\u2000` na `\u200A`
  - `\u2028`
  - `\u2029`
  - `\u202F`
  - `\u205F`
  - `\u3000`
- Výrazy:
  - Selektor je podmíněný výraz. Selektor, který se vyhodnocuje na skutečné shody; selektor, který se vyhodnocuje na hodnotu false nebo neznámý, se neshoduje.
  - Aritmetické výrazy se skládají z sebe, aritmetických operací, identifikátorů (hodnota identifikátoru je považována za číselný literál) a numerické literály.
  - Podmíněné výrazy se skládají z vlastních, porovnávacích operací a logických operací.
- Standardní bracketing () pro nastavení pořadí, ve kterém jsou výrazy vyhodnocovány, je podporován.
- Logické operátory v pořadí priority: `NOT`, `AND`, `OR`.
- Operátory porovnání: =, >, >=, <, <=, <> (nerovná se).
  - Dva bajtové řetězce se rovnají, pouze pokud řetězce mají stejnou délku a pořadí bajtů je stejné.
  - Porovnávají se pouze hodnoty stejného typu. Výjimkou je, že je platný pro porovnání přesných číselných hodnot a přibližné numerické hodnoty (typ požadované konverze je definován pravidly pro numerickou propagaci Java). Pokud dojde k pokusu o porovnání různých typů, je selektor vždy nepravdivý.
  - Porovnání řetězců a logických hodnot je omezeno na = a <>. Dva řetězce jsou shodné pouze tehdy, obsahují-li stejnou posloupnost znaků.
- Aritmetické operátory v pořadí priority:
  - +, - unární.
  - Rozdělení * násobení a / .
  - + sčítání a odčítání - .
  - Aritmetické operace na hodnotě NULL nejsou podporovány. Pokud se o tyto pokusy pokusí, úplný selektor je vždy nepravdivý.
  - Aritmetické operace musí používat numerickou propagaci Java.
- arithmetic-expr1 [`NOT`] `BETWEEN` arithmetic-expr2 a arithmetic-expr3 operátor porovnání:
  - `Age BETWEEN 15 and 19` je ekvivalentní s `age >= 15 AND age <= 19`.

  - `Age NOT BETWEEN 15 and 19` je ekvivalentní s `age < 15 OR age > 19`.
  - Má-li některý z výrazů operace `BETWEEN` hodnotu `NULL`, je hodnota operace false. Je-li některý z výrazů operace `NOT BETWEEN` `NULL`, hodnota operace je true.
- Identifikátor `[NOT] IN (string-literal1, string-literal2,...)` operátoru porovnání, kde identifikátor má hodnotu typu String nebo `NULL` .
  - `Country IN ('UK', 'US', 'France')` je true pro `'UK'` a false pro `'Peru'`. Je ekvivalentní výrazu `(Country = 'UK') OR (Country = 'US') OR (Country = 'France')`.
  - `Country NOT IN ('UK', 'US', 'France')` je false pro `'UK'` a true pro `'Peru'`. Je ekvivalentní výrazu `NOT ((Country = 'UK') OR (Country = 'US') OR (Country = 'France'))`.
  - Je-li identifikátor operace `IN` nebo `NOT IN` `NULL`, hodnota operace je neznámá.
- `identifier [NOT] LIKE` *`pattern-value`* `[ESCAPE` *`escape-character`*`]` porovnávacího operátoru, kde `identifier` má řetězcovou hodnotu. *pattern-value* je řetězcový literál, kde _ zastupuje libovolný jednotlivý znak a % zastupuje libovolnou posloupnost znaků (včetně prázdné posloupnosti). Všechny ostatní znaky jsou stojan pro sebe. Volitelný řídicí znak *escape-character* je jednoznakový řetězcový literál, který se používá k úniku speciálního významu _ a % v *vzor-vzor*. Operátor `LIKE` musí být použit pouze k porovnání dvou řetězcových hodnot.
  - `phone LIKE '12%3'` je true pro 123 a 12993 a false pro 1234.
  - `word LIKE 'l_se'` je true pro ztrátu a false pro uvolnění.
  - `underscored LIKE '\_%' ESCAPE '\'` je true pro `_foo` a false pro `bar`.
  - `phone NOT LIKE '12%3'` je false pro 123 a 12993 a true pro 1234.
  - Je-li identifikátor operace `LIKE` nebo `NOT LIKE` `NULL`, hodnota operace je neznámá.

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

  Vyhodnocení výrazů selektoru, které obsahují hodnoty `NULL` , je definováno sémantikou SQL 92 `NULL` , v souhrnu:
  - SQL považuje hodnotu `NULL` za neznámou.
  - Porovnání nebo aritmetika s neznámou hodnotou vždy poskytne neznámou hodnotu.
  - Operátory `IS NULL` a `IS NOT NULL` převádějí neznámou hodnotu na hodnoty `TRUE` a `FALSE` .

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

Tabulka 1. Výsledek logického operátoru, je-li logika A AND B

Operátor A	**Operátor B**	**Výsledek (A AND B)**
T	F	F
T	U	U
T	T	T
F	T	F
F	U	F

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

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

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

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

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

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

Komentáře SQL nejsou podporovány.

Související pojmy

Chování výběru
Přehled chování výběru produktu IBM WebSphere MQ .

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

“Vlastnosti zprávy” na stránce 17

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

Související odkazy

MsgHandle

MQBUFMH-Převedení vyrovnávací paměti na popisovač zprávy

Pravidla řetězce výběru a omezení

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

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

Související pojmy

Aspekty UTF-8 a Unicode při použití selektorů zpráv

Aspekty UTF-8 a Unicode při použití selektorů zpráv

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

Názvy vlastností zprávy mohou být zadány pomocí libovolné platné posloupnosti znaků Unicode. Názvy vlastností zprávy obsažené ve výběrových řetězcích, které jsou kódovány v UTF-8 , budou ověřeny i v případě, že obsahují vícebajtové znaky. Ověření vícebajtové znakové sady UTF-8 je striktní a musíte zajistit, aby pro názvy vlastností zpráv byly použity platné posloupnosti UTF-8 .

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

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

Související pojmy

Pravidla řetězce výběru a omezení

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

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

Když aplikace publikuje na řetězec tématu, kde jeden nebo více odběratelů má výběrový řetězec výběru obsahu zprávy, produkt WebSphere MQ bude požadovat, aby poskytovatel výběru rozšířených zpráv zanalyzovat publikování a informovat produkt WebSphere MQ o tom, zda publikování odpovídá kritériím výběru určeným každým odběratelem s filtrem obsahu.

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

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

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

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

Související pojmy
Chování výběru
Přehled chování výběru produktu IBM WebSphere MQ .

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

Asynchronní spotřeba zpráv produktu IBM WebSphere MQ

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

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

V asynchronní spotřebě zpráv se používají následující termíny:

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

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

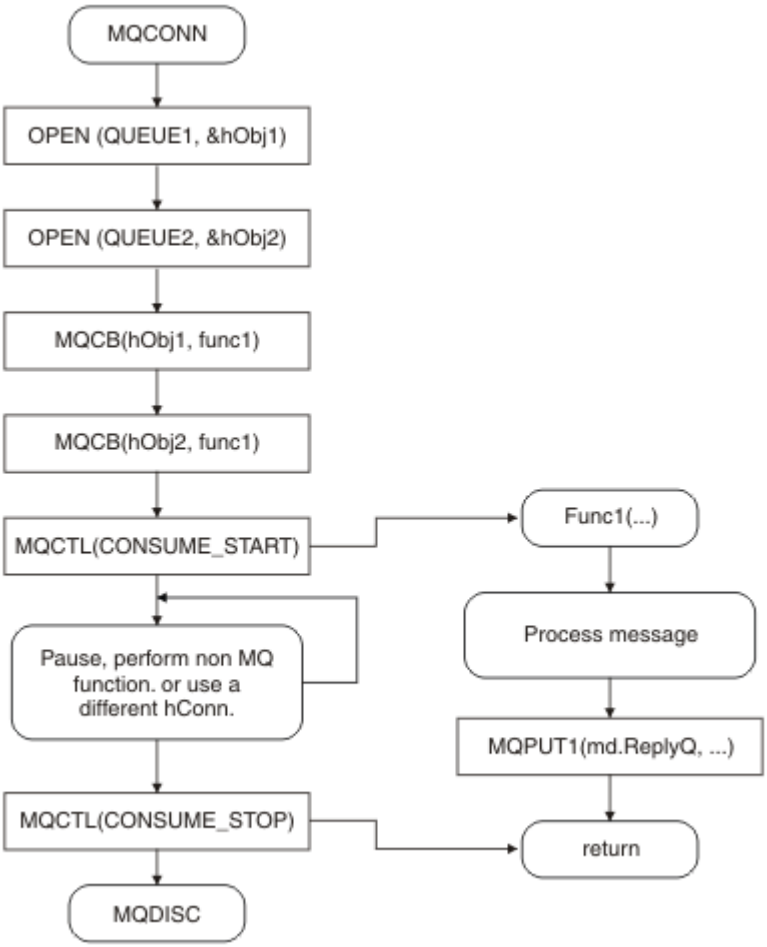
Zpětné volání
Generický termín používaný k odkazování na rutinu Message Consumer nebo obslužné rutiny událostí.

Asynchronní spotřeba může zjednodušit návrh a implementaci nových aplikací, zejména těch, které zpracovávají více vstupních front nebo odběrů. Pokud však používáte více než jednu vstupní frontu a zpracovával zprávy v posloupnosti priorit, je posloupnost priorit sledována nezávisle v každé frontě:

můžete dostat zprávy s nízkou prioritou z jedné fronty před zprávami s vysokou prioritou od druhé. Pořadí zpráv v rámci více front není garantováno. Všimněte si také, že pokud používáte uživatelské procedury rozhraní API, může být zapotřebí, abyste je změnili, aby zahrnovaly volání MQCB a MQCTL.

Následující ilustrace poskytují příklad, jak můžete tuto funkci použít.

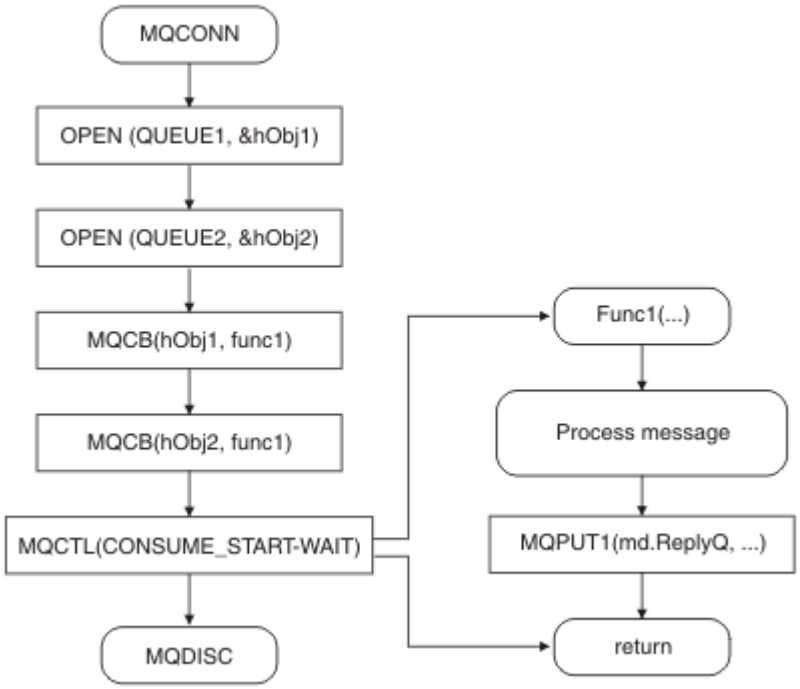
Produkt Obrázek 5 na stránce 33 zobrazuje zprávy s vícenásobnými aplikacemi spotřebovávající zprávy ze dvou front. Příklad zobrazuje všechny zprávy doručené do jedné funkce.



Obrázek 5. Standardní aplikace řízená zprávami spotřebovávající ze dvou front

Obrázek 6 na stránce 34 Tento ukázkový tok zobrazuje jednovláknovou aplikaci spotřebovávající zprávy ze dvou front. Příklad zobrazuje všechny zprávy doručené do jedné funkce.

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



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

Skupiny zpráv

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

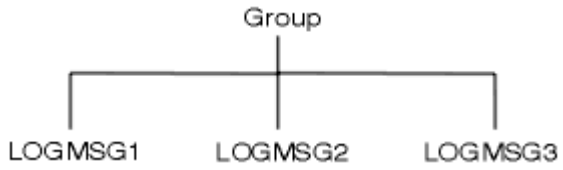
Skupiny zpráv umožňují označit více zpráv jako vzájemně související a logický příkaz, který má být použit ke skupině (viz “Logické a fyzické uspořádání” na stránce 236). Na jiných platformách než z/OSumožňuje produkt “Segmentace zpráv” na stránce 252 rozdělit velké zprávy na menší segmenty. Seskupené nebo segmentované zprávy nelze použít při vkládání do tématu.

Hierarchie v rámci skupiny je následující:

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

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

Obrázek 7 na stránce 34 zobrazuje skupinu logických zpráv:



Obrázek 7. Skupina logických zpráv

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

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

Použít logické zprávy v rámci skupiny k:

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

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

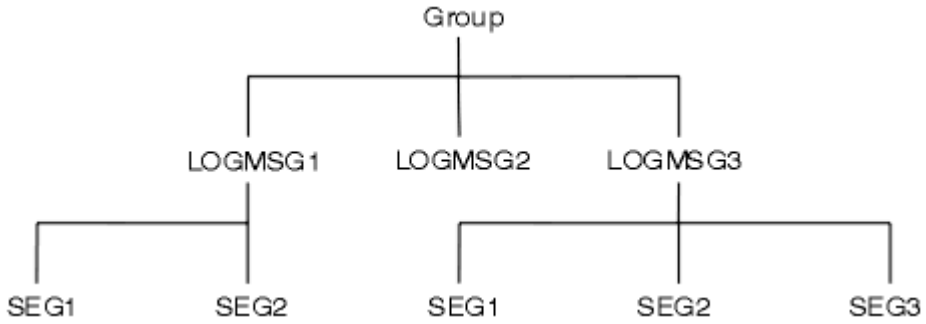
Segment

Segmenty se používají ke zpracování zpráv, které jsou příliš velké pro vložení nebo získání aplikace nebo správce front (včetně zasahujících správců front, kterými se zpráva předává). Další informace viz "Segmentace zpráv" na stránce 252.

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

Každý segment se skládá z jedné fyzické zprávy, která může patřit do skupiny (Obrázek 8 na stránce 35 ukazuje příklad zpráv uvnitř skupiny). Segment je logicky součástí jediné zprávy, takže pouze pole *MsgId*, *Offset*a *SegmentFlag* v deskriptoru MQMD by se měly lišit mezi oddělenými segmenty stejné zprávy. Pokud se nezdaří doručení segmentu, vrátí se podle potřeby kód příčiny MQRC_INCOMPLEE_GROUP nebo MQRC_INCOMPLEE_MSG .

Obrázek 8 na stránce 35 zobrazuje skupinu logických zpráv, z nichž některé jsou segmentovány:



Obrázek 8. Segmentované zprávy

Segmentované nebo seskupené zprávy nemůžete používat s publikováním/odběrem.

Popis logických a fyzických zpráv naleznete v části "Logické a fyzické uspořádání" na stránce 236. Další informace o segmentování zpráv viz "Segmentace zpráv" na stránce 252.

Trvalost zpráv

Trvalé zprávy se zapisují do protokolů a datových souborů fronty.

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

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

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

- Trvalá zpráva, mimo aktuální jednotku práce, se zapisuje na disk při každé operaci put a get. Viz "Potvrzení a zálohování jednotek práce" na stránce 310.

- V produktu IBM WebSphere MQ na systémech UNIX , IBM WebSphere MQ na systémech Linux , a IBM WebSphere MQ pro produkt Windowsse trvalá zpráva v rámci aktuální jednotky práce protokoluje pouze tehdy, když je jednotka práce potvrzena (a jednotka práce může obsahovat mnoho operací fronty).

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

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

Zprávy, které se nepodařilo doručit

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

Můžete provést následující akce:

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

Další informace viz “Obsluha chyb programu” na stránce 529.

Zprávy, které jsou zálohovány

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

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

V produktu WebSphere MQ pro WebSphere MQ for Windows, WebSphere MQ na systémech UNIX , WebSphere MQ na systémech Linux počet vrácení vždy přežije restarty správce front. Jakákoli změna atributu *HardenGetBackout* se ignoruje.

Další informace o potvrzování a vracení zpráv naleznete v tématu “Potvrzení a zálohování jednotek práce” na stránce 310.

Fronta pro zařazení do fronty a správce front

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

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

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

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

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

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

Poznámka: Můžete požádat správce front o více pokusů o doručení zprávy a můžete požádat, aby byla zpráva vyřazena, pokud selže. Není-li zpráva po selhání k doručení vyřazena, vzdálený správce front vloží zprávu do fronty nedoručených zpráv (viz "Použití fronty nedoručených zpráv (nedoručená zpráva)" na stránce 532).

kontext zprávy

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

Načítání aplikace může vyžadovat:

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

Použijete-li volání MQPUT nebo MQPUT1 k vložení zprávy do fronty, můžete určit, že správce front má přidat některé výchozí kontextové informace do deskriptoru zpráv. Aplikace, které mají odpovídající úroveň oprávnění, mohou přidávat další informace o kontextu. Další informace o tom, jak zadat informace o kontextu, viz "Řízení informací o kontextu" na stránce 222.

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

- Potvrdit při doručení
- Vypršení

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

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

kontext identity

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

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

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

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

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

Původní kontext

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

PutApplType	Typ aplikace, která vložila tuto zprávu (například transakce CICS).
PutApplName	Název aplikace, která vložila tuto zprávu (například, název úlohy nebo transakce).
PutDate	Datum, kdy byla zpráva vložena do fronty.
PutTime	Čas, kdy byla zpráva vložena do fronty.
ApplOriginData	Jakékoli další informace, které chce aplikace zahrnout o původu zprávy. Může být například nastaven vhodně autorizovanými aplikacemi, které označují, zda jsou data identity důvěryhodná.

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

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

Objekty produktu WebSphere MQ

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

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

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

Související pojmy

“Řízení informací o kontextu” na stránce 222
Použijete-li volání MQPUT nebo MQPUT1 k vložení zprávy do fronty, můžete určit, že správce front má přidat některé výchozí kontextové informace do deskriptoru zpráv. Aplikace, které mají odpovídající úroveň oprávnění, mohou přidávat další informace o kontextu. Pole voleb ve struktuře MQPMO můžete použít k řízení informací o kontextu.

Související odkazy

“Volby MQOPEN vztahující se ke kontextu zprávy” na stránce 214
Chcete-li mít možnost přidružit kontextové informace ke zprávě, když ji vložíte do fronty, musíte při otevření fronty použít jednu z voleb kontextu zprávy.

Příprava a spuštění aplikací Microsoft Transaction Server

Chcete-li připravit aplikaci MTS ke spuštění jako aplikaci klienta WebSphere MQ MQI, postupujte podle těchto pokynů, které odpovídají vašemu prostředí.

Obecné informace o vývoji aplikací serveru Microsoft Transaction Server (MTS), které přistupují k prostředkům produktu WebSphere MQ , naleznete v části týkající se MTS v centru nápovědy WebSphere MQ .

Chcete-li připravit aplikaci MTS ke spuštění jako aplikaci klienta WebSphere MQI MQ , proveďte jednu z následujících možností pro každou komponentu aplikace:

- Pokud komponenta používá vazby jazyka C pro rozhraní MQI, postupujte podle pokynů v části "Příprava programů jazyka C v systému Windows" na stránce 443 , ale propojte ji s knihovnou mqicxa.lib místo mqic.lib.
- Pokud komponenta používá třídy jazyka C++ produktu WebSphere MQ , postupujte podle pokynů v části "Sestavování programů C++ v systému Windows" na stránce 631 , ale propojte ji s knihovnou imqx23vn.lib namísto imqc23vn.lib.
- Pokud komponenta používá vazby jazyka Visual Basic pro rozhraní MQI, postupujte podle pokynů v příručce "Příprava programů jazyka Visual Basic v systému Windows" na stránce 447 , ale při definování projektu Visual Basic zadejte do pole **Argumenty podmíněné kompilace** hodnotu `MqType=3` .
- Pokud komponenta používá třídy automatizace produktu WebSphere MQ pro ActiveX (MQAX), definujte proměnnou prostředí GMQ_MQ_LIB s hodnotou `mqic32xa.dll` .

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

Použití IBM WebSphere MQ s WebSphere Application Server

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

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

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

Aplikace mohou pro objekt továrny připojení fronty použít buď *přenos klienta* , nebo *přenos vazeb* . Pro *přenos vazeb* musí správce front existovat lokálně na aplikaci, která vyžaduje připojení. Pokud správce front není lokální vzhledem k aplikaci, měla by být nainstalována aplikace *Připojení klienta* , která umožní aplikaci připojit se ke správci front spuštěnému v jiném počítači nebo obrazu.

Ve výchozím nastavení služba JMS zadržená ve frontách produktu IBM WebSphere MQ používá záhlaví MQRFH2 k ukládání některých informací záhlaví zprávy JMS. Mnoho starších aplikací produktu IBM WebSphere MQ nemůže zpracovávat zprávy s těmito záhlavími a vyžadovat jejich vlastní záhlaví charakteristik, například MQCIH pro most CICS Bridge nebo MQWIH pro aplikace IBM WebSphere MQ Workflow. Další podrobnosti o těchto speciálních aspektech viz "Mapování zpráv JMS na zprávy produktu WebSphere MQ" na stránce 784.

Scénáře transakčního podpory

Použití podpory transakcí umožňuje vašim aplikacím spolehlivě pracovat s databázemi.

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Tento oddíl představuje podporu transakcí. Práce vyžadovaná k povolení aplikací pro použití produktu IBM WebSphere MQ s databázovým produktem zahrnuje oblasti programování aplikací a správy systému. Tyto informace použijte spolu s "Potvrzení a zálohování jednotek práce" na stránce 310.

Začneme tím, že zavedeme jednotky práce, které tvoří transakce, pak popište způsoby, jak produkt IBM WebSphere MQ umožňuje koordinovat transakce s databázemi.

Související pojmy

"Představení pracovních jednotek" na stránce 40
Toto téma představuje a definuje obecné koncepce jednotky práce, potvrzení, odvolání a synchronizace. Obsahuje také dva scénáře, které ilustrují globální pracovní jednotky.

IBM WebSphere MQ a HP NonStop TMF

Představení pracovních jednotek

Toto téma představuje a definuje obecné koncepce jednotky práce, potvrzení, odvolání a synchronizace. Obsahuje také dva scénáře, které ilustrují globální pracovní jednotky.

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Když program vloží zprávy do front v rámci pracovní jednotky, tyto zprávy se zviditelní pro ostatní programy pouze v případě, že program *potvrdí* jednotku práce. Chcete-li potvrdit jednotku práce, musí být všechny aktualizace úspěšné, aby se zachovala integrita dat.

Pokud program zjistí chybu a rozhodne se neprovádět operaci vložení jako permanentní, může ji *zálohovat* jednotku práce. Když program provádí odvolání, produkt WebSphere MQ obnoví fronty odebráním zpráv, které byly vloženy do front touto jednotkou práce.

Podobně, když program získává zprávy z jedné nebo více front v rámci pracovní jednotky, tyto zprávy zůstanou ve frontách, dokud program nepotvrdí jednotku práce, ale zprávy nejsou k dispozici pro načtení jinými programy. Zprávy jsou trvale odstraněny z front, když program potvrdí jednotku práce. Pokud program zálohuje pracovní jednotku, produkt WebSphere MQ obnoví fronty tím, že zpřístupní zprávy, které mají být načteny jinými programy.

Rozhodnutí o potvrzení nebo vrácení změn se provádí v nejjednodušším případě na konci úlohy. Může však být užitečnější, aby aplikace synchronizovala změny dat v jiných logických bodech v rámci úlohy. Tyto logické body se nazývají synchronizační body (nebo synchronizační body) a období zpracování sady aktualizací mezi dvěma body synchronizace se nazývá *jednotka práce*. Několik volání MQGET a volání MQPUT může být součástí jediné jednotky práce.

S produktem WebSphere MQje třeba rozlišovat mezi *lokálními* a *globálními* jednotkami práce:

Místní jednotky práce

Jedná se o ty, ve kterých jsou povoleny pouze jediné akce, a získat z front WebSphere MQ a koordinace každé jednotky práce je poskytována ve správci front pomocí procesu *jednofázového potvrzení* .

Lokální jednotky práce použijte v případě, že jediným prostředkem, který má být aktualizován, jsou fronty, které jsou spravovány jedním správcem front WebSphere MQ . Aktualizace se potvrdí pomocí příkazu MQCMIT nebo jsou vráceny pomocí operace MQBACK.

K dispozici nejsou žádné úlohy správy systému, kromě správy protokolů, které se podílejí na používání lokálních transakcí. Ve vašich aplikacích použijte volání MQPUT a MQGET s MQCMIT a MQBACK, zkuste použít volby MQPMO_SYNCPOINT a MQGMO_SYNCPOINT. (Informace o správě protokolů viz Správa souborů protokolu .)

Globální pracovní jednotky

Jsou aktualizovány také další prostředky, jako jsou například tabulky v relační databázi. Existuje-li více než jeden *správce prostředků* , je zapotřebí softwaru *správce transakcí* , který používá proces *dvoufázového potvrzování* ke koordinaci globální transakce.

Globální pracovní jednotky použijte v případě, že potřebujete zahrnout také aktualizace do softwaru správce relačních databází, jako jsou Db2, Oracle, Sybasea Informix.

Pro použití globálních pracovních jednotek existuje několik možných scénářů. Dokumentovaný zde jsou dva scénáře:

1. V první řadě samotný správce front vystupuje jako správce transakcí. V tomto scénáři příkazy MQI řídí globální jednotky práce. Jsou spouštěny v aplikacích pomocí příkazového slova MQBEGIN a poté jsou potvrzeny pomocí MQCMIT nebo byly vráceny pomocí operace MQBACK.
2. Ve druhé roli je role správce transakcí prováděna jiným softwarem, jako například TXSeries, Encinanebo Tuxedo. V tomto scénáři je rozhraní API poskytované softwarem správce transakcí použito k řízení jednotky práce (například EXEC CICS SYNCPOINT for TXSeries).

Následující části popisují všechny kroky nezbytné k použití globálních pracovních jednotek uspořádané podle těchto dvou scénářů:

- "Scénář 1: Správce front provádí koordinaci" na stránce 41
- "Scénář 2: Další software poskytuje koordinaci" na stránce 66

Scénář 1: Správce front provádí koordinaci

Ve scénáři 1 se správce front chová jako správce transakcí. V tomto scénáři příkazy MQI řídí globální jednotky práce. Jsou spouštěny v aplikacích pomocí příkazového slova MQBEGIN a poté jsou potvrzeny pomocí MQCMIT nebo byly vráceny pomocí operace MQBACK.

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Úroveň izolace

V produktu IBM WebSphere MQmůže být zpráva ve frontě viditelná před aktualizací databáze v závislosti na návrhu izolace transakce implementovaném v rámci databáze.

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Pokud správce front produktu IBM WebSphere MQ pracuje ve funkci správce transakcí standardu XA pro koordinaci aktualizací pro správce prostředků XA, postupuje se podle následujícího protokolu potvrzení:

1. Připravte všechny správce prostředků XA.
2. Potvrďte správce prostředků správce front produktu IBM WebSphere MQ .
3. Potvrdit další správce prostředků.

Mezi krokem 2 a 3 může aplikace zobrazit zprávu, která je potvrzena do fronty, ale odpovídající řádek v databázi tuto zprávu neodráží.

Nejedná se o problém, je-li databáze konfigurována tak, že volání rozhraní API databáze aplikace čeká na dokončení nevyřízených aktualizací.

Tento problém můžete vyřešit konfigurací databáze jiným způsobem. Typ potřebného nastavení konfigurace je označován jako "úroveň oddělení". Další informace o úrovních oddělení naleznete v dokumentaci k databázi. Správce front můžete alternativně nakonfigurovat tak, aby správce prostředků byl schopen potvrdit následující obrácené pořadí:

1. Připravte všechny správce prostředků XA.
2. Potvrdit další správce prostředků.
3. Potvrďte správce prostředků správce front produktu IBM WebSphere MQ .

Při změně protokolu se správce front produktu IBM WebSphere MQ potvrdí jako poslední, takže aplikace, které čtou zprávy z front, uvidí zprávu až po dokončení příslušné aktualizace databáze.

Chcete-li nakonfigurovat správce front tak, aby používal tento změněný protokol, nastavte proměnnou prostředí **AMQ_REVERSE_COMMIT_ORDER** .

Nastavte tuto proměnnou prostředí v prostředí, ze kterého má být spuštěn produkt **strmqm** ke spuštění správce front. Například před spuštěním správce front spusťte následující příkaz v shellu:

```
export AMQ_REVERSE_COMMIT_ORDER=1
```

Poznámka: Nastavení této proměnné prostředí může způsobit další položku protokolu na transakci, takže to bude mít malý dopad na výkon každé transakce.

Koordinace databáze

Když správce front koordinuje globální jednotky práce samotné, je možné integrovat aktualizace databáze v rámci jednotek práce. To znamená, že lze zapsat smíšenou aplikaci MQI a SQL a lze použít příkazy MQCMIT a MQBACK k potvrzení nebo odvolání změn ve frontách a databázích společně.

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Správce front toho dosáhne pomocí protokolu dvoufázového potvrzování, který je popsán v tématu *Zpracování distribuovaných transakcí X/Open: Specifikace XA*. Má-li být jednotka práce potvrzena, správce front se nejprve dotáže každého správce databází, zda je připraven potvrdit své aktualizace. Všechny aktualizace fronty a databáze jsou potvrzeny pouze v případě, že jsou všechny účastníky, včetně samotného správce front, připraveny k potvrzení. Pokud žádný účastník nemůže připravit své aktualizace, bude místo toho odvrácena jednotka práce.

Obecně platí, že globální pracovní jednotka je implementována v aplikaci následující metodou (v pseudokódu):

```
MQBEGIN
MQGET (včetně příznaku MQGMO_SYNCPOINT v rámci voleb zprávy)
MQPUT (uveďte příznak MQPMO_SYNCPOINT ve volbách zprávy)
VLOŽENÍ SQL
MQCMIT
```

Účelem operace MQBEGIN je označení začátku globální transakce. Cílem MQCMIT je označení konce globální pracovní jednotky a dokončit ji se všemi zúčastněnými správci prostředků pomocí protokolu dvoufázového potvrzování.

Je-li transakce (také známá jako *transakce*) úspěšně dokončena pomocí příkazu MQCMIT, všechny akce provedené v rámci této jednotky práce jsou trvalé nebo nevratné. Pokud z jakéhokoli důvodu jednotka práce selže, všechny akce se místo toho odzálohují. Není možné, aby jedna akce v pracovní jednotce byla trvale prováděna, zatímco jiná je vrácena. Jedná se o princip pracovní jednotky: buď všechny akce v rámci pracovní jednotky jsou trvalé nebo žádné z nich nejsou.

Poznámka:

1. Aplikační programátor může vynutit, aby byla jednotka práce vrácena voláním funkce MQBACK. Pokud se aplikace nebo databáze *nezdaří* před vyvoláním funkce MQCMIT, je správce front také odvolán správcem front.
2. Pokud aplikace volá funkci MQDISC bez volání MQCMIT, bude se správce front chovat, jako by byla volána funkce MQCMIT, a potvrzuje jednotku práce.

Ve struktuře mezi MQBEGIN a MQCMIT správce front nevolá žádné volání do databáze, aby aktualizoval své prostředky. To znamená, že jediným způsobem, jak jsou tabulky databáze změněny, je váš kód (například SQL INSERT v pseudokódu).

Podpora úplného obnovení je poskytována v případě, že správce front ztratí kontakt s některým ze správců databází během potvrzování protokolu. Pokud bude správce databází v nejistém stavu nedostupný, je úspěšně připraven potvrdit, ale přesto má přijmout rozhodnutí o potvrzení nebo odvolání, správce front si zapamatuje výsledek transakce, dokud tento výsledek nebude úspěšně doručen do databáze. Podobně platí, že je-li správce front ukončen s neprovedenými neúplnými operacemi operace commit, jsou tyto operace zapamatovány při restartování správce front. Pokud dojde k neočekávanému ukončení aplikace,

integrita jednotky práce není ohrožena, ale výsledek závisí na tom, kde v procesu byla aplikace ukončena, jak je popsáno v tématu Tabulka 5 na stránce 43.

Co se stane, když dojde k selhání databáze nebo aplikačního programu, v následujících tabulkách:

Tabulka 4. Co se stane, když dojde k selhání databázového serveru

Výskyt selhání	**Výsledek**
Před voláním aplikace MQCMIT.	Jednotka práce je zálohována.
Během volání aplikace na MQCMIT, **před** všemi databázemi označila, že jsou úspěšně připraveny.	Transakce je zálohována s kódem příčiny MQRC_BACKED_OUT.
Během volání aplikace MQCMIT **po** všechny databáze indikovala, že se úspěšně připravili, ale před tím, než všichni uvedli, že se úspěšně zavázali.	Jednotka práce je zadržena správcem front v zotavitelném stavu s kódem příčiny MQRC_OUTCOME_PENDING.
Během volání aplikace na MQCMIT **po** všechny databáze indikovala, že se úspěšně dopustily.	Jednotka práce je potvrzena s kódem příčiny MQRC_NONE.
Po volání aplikace MQCMIT.	Jednotka práce je potvrzena s kódem příčiny MQRC_NONE.

Tabulka 5. Co se stane, když selže aplikační program

Výskyt selhání	**Výsledek**
Před voláním aplikace MQCMIT.	Jednotka práce je zálohována.
Během volání aplikace na MQCMIT, **před** správcem front byl přijat požadavek MQCMIT aplikace.	Jednotka práce je zálohována.
Během volání aplikace na MQCMIT, **za** správce front přijal požadavek MQCMIT aplikace.	Správce front se pokouší potvrdit použití dvoufázového potvrzování (s výhradou databázových produktů úspěšně prováděných a potvrzujících jejich součástí jednotky práce).

V případě, že je kód příčiny na vratku z MQCMIT MQRC_OUTCOME_PENDING, správce front zapamatuje pracovní jednotku, dokud nebude moci znovu navázat spojení s databázovým serverem a nepotvrdí její část jednotky práce. Informace o tom, jak a kdy bylo provedeno zotavení, naleznete v části “Aspekty při ztrátě kontaktu se správcem prostředků XA” na stránce 59 .

Správce front komunikuje se správci databází pomocí rozhraní XA, jak je popsáno v tématu *Zpracování distribuovaného zpracování transakcí X/Open: Specifikace XA*. Příklady těchto volání funkce jsou xa_open, xa_start, xa_end, xa_prepare a xa_commit. Termíny *transaction manager* a *resource manager* používáme ve stejném smyslu, jako jsou použity ve specifikaci XA.

Omezení
Pro podporu koordinace databází existují omezení.

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Platí následující omezení:

- Schopnost koordinovat aktualizace databáze v rámci jednotek práce produktu WebSphere MQ je **ne** podporována v aplikaci klienta MQI. Použití operace MQBEGIN v aplikaci klienta selže. Program, který volá funkci MQBEGIN, musí být spuštěn jako aplikace *server* na stejném počítači jako správce front.

  Poznámka: Aplikace *server* je program, který byl propojen s nezbytnými knihovnami serveru WebSphere MQ ; aplikace *client* je program, který byl propojen s nezbytnými knihovnami klienta WebSphere MQ . Podrobnosti o kompilaci a propojení vašich programů viz “Sestavování aplikací pro

klienty WebSphere MQ MQI” na stránce 342 a “Sestavení aplikace IBM WebSphere MQ” na stránce 412 .

- Databázový server může být umístěn na jiném počítači než server správce front, pokud je databázový klient nainstalován na stejném počítači jako správce front, a podporuje tuto funkci. Podívejte se do dokumentace databázového produktu a zjistěte, zda je možné použít klientský software pro systémy s dvoufázovým potvrzováním.
- Ačkoli se správce front chová jako správce prostředků (pro účely, aby se zapojil do globálních pracovních jednotek scénáře 2), není možné vytvořit jednoho správce front koordinovaného jiného správce front v rámci globálních pracovních jednotek ve scénáři 1.

Zaměnit zaváděcí soubory

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Zaváděcí soubor přepínače je sdílená knihovna (knihovna DLL v systému Windows) načtená kódem ve vaší aplikaci IBM WebSphere MQ a správci front. Jeho účelem je zjednodušit načítání sdílené knihovny klienta databáze a vrátit ukazatele na funkce XA.

Před spuštěním správce front musí být uvedeny podrobnosti o souboru načtení přepínače. Podrobnosti jsou umístěny do souboru qm.ini v systému Windows, systémy UNIX and Linux .

- V systémech Windows a Linux (platformyx86 a x86-64) aktualizujte soubor qm.ini pomocí produktu IBM WebSphere MQ Explorer .
- Na všech ostatních systémech upravte soubor qm.inipřímo.

Zdroj jazyka C pro soubor načtení přepínače se dodává spolu s instalací produktu IBM WebSphere MQ , pokud podporuje globální pracovní jednotky scénáře 1. Zdroj obsahuje funkci s názvem MQStart. Po načtení zaváděcího souboru přepínače vyvolá správce front tuto funkci, která vrátí adresu struktury s názvem *přepínač XA*.

Struktura přepínače XA existuje ve sdílené knihovně klienta databáze a obsahuje několik ukazatelů funkcí, jak je popsáno v tématu Tabulka 6 na stránce 44:

Tabulka 6. Ukazatele funkce přepínače XA

Název ukazatele funkce	**funkce XA**	**Účel**
xa_open_entry	xa_open	Připojit se k databázi
položka xa_close_entry	xa_close	Odpojit od databáze
za_vstup_	xa_start	Spuštění větve globální jednotky práce
xa_end_	xa_end	Pozastavit větev globální pracovní jednotky
xa_rollback_entry	xa_rollback	Odvolat větev globální pracovní jednotky
xa_prepare_entry	xa_připravit	Příprava na potvrzení větve globální jednotky práce
xa_commit_entry	xa_commit	Potvrdit větev globální pracovní jednotky
xa_recover_entry	xa_obnovit	Zjistit z databáze, zda má neověřnou jednotku práce
xa_forget_entry	xa_zapomenout	Povolit databázi zapomenout na větev globální jednotky práce

Tabulka 6. Ukazatele funkce přepínače XA (pokračování)		
Název ukazatele funkce	**funkce XA**	**Účel**
xa_complete_entry	xa_complete	Dokončit větev globální pracovní jednotky

Během prvního volání MQBEGIN ve vaší aplikaci načte kód IBM WebSphere MQ , který se provádí jako součást operace MQBEGIN, načtený zaváděcí soubor přepínače a vyvolá funkci xa_open ve sdílené knihovně databáze. Podobně při spuštění správce front a při dalších následných příležitostech některé procesy správce front načtou zaváděcí soubor přepnutí a volání xa_open.

Počet volání xa_ * můžete snížit pomocí *dynamické registrace*. Úplný popis této techniky optimalizace naleznete v tématu "Dynamická registrace XA" na stránce 64.

Konfigurace systému pro koordinaci databáze

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Existuje několik úloh, které je třeba provést před tím, než se správce databází může účastnit globálních pracovních jednotek koordinovaných správcem front. Zde jsou popsány následující informace:

- "Instalace a konfigurace databázového produktu" na stránce 45
- "Vytvoření souborů načtení přepínače" na stránce 46
- "Přidání informací o konfiguraci do správce front" na stránce 46
- "Psaní a úprava vašich aplikací" na stránce 48
- "Testování systému" na stránce 48

Instalace a konfigurace databázového produktu
Chcete-li nainstalovat a nakonfigurovat databázový produkt, prohlédněte si vlastní dokumentaci produktu. Tato témata v tomto oddílu popisují obecné problémy konfigurace a informace o tom, jak souvisejí s interoperami mezi produktem WebSphere MQ a databází.

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Databázová připojení

Aplikace, která ustanovuje standardní připojení ke správci front, je přidružena k podprocesu v samostatném procesu agenta lokálního správce front. (Připojení, které není cestou *fastpath* , je v tomto kontextu *standardní* připojení. Další informace viz "Připojení ke správci front pomocí volání MQCONNX" na stránce 201.)

Když aplikace vydá `MQBEGIN`, obě tato data a proces agenta volají funkci xa_open v knihovně klienta databáze. V reakci na tuto skutečnost se kód knihovny klienta databáze *připojuje* k databázi, která se má podílet na transakci *jak z procesů aplikace, tak z procesů správce front*. Tato databázová připojení zůstanou zachována po dobu, po kterou bude aplikace nadále připojena ke správci front.

Jedná se o důležitou úvahu, pokud databáze podporuje pouze omezený počet uživatelů nebo připojení, protože se v databázi provádí dvě připojení na podporu jednoho aplikačního programu.

Konfigurace klienta/serveru

Knihovna klienta databáze, která je načtena do správce front WebSphere MQ a aplikační procesy **musí** být schopna odesílat a přijímat od serveru. Ujistěte se, že platí:

- Konfigurační soubory klienta/serveru databáze mají správné podrobnosti.
- Příslušné proměnné prostředí jsou nastaveny v prostředí správce front **a** aplikačních procesů.

Vytvoření souborů načtení přepínače
Produkt WebSphere MQ se dodává s ukázkovým souborem Makefile, který slouží k sestavení souborů načtení přepínače pro podporované správce databází.

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

Ukázkový soubor Makefile spolu se všemi přidruženými zdrojovými soubory jazyka C vyžadovanými pro sestavení souborů načtení přepínače je nainstalován v následujících adresářích:

- Pro produkt WebSphere MQ for Windows, v adresáři *MQ_INSTALLATION_PATH*\tools\c\samples\xatm\
- Pro systémy WebSphere MQ for UNIX and Linux , v adresáři *MQ_INSTALLATION_PATH*/samp/xatm/

Ukázkové zdrojové moduly použité k sestavení souborů načtení přepínače jsou:

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

Při generování souborů načtení přepínače nainstalujte 32bitové soubory načtení přepínače do produktu /var/mqm/exits a v produktu /var/mqm/exits64nainstalujte 64bitové zaváděcí soubory přepínače.

Pokud máte 32bitového správce front, ukázkový soubor make, xaswit.mak, nainstaluje 32bitový soubor LOAD s 32bitovým přepínačem do produktu /var/mqm/exits.

Pokud máte 64bitové správce front, ukázkový soubor make, xaswit.mak, nainstaluje 32bitový soubor načtení přepínače do /var/mqm/exitsa do 64bitového zaváděcího souboru přepínače /var/mqm/exits64.

Zabezpečení souborů

Je možné, že operační systém selže při načítání zaváděcího souboru přepínače produktem WebSphere MQ, a to z důvodů, které nejsou mimo kontrolu produktu WebSphere MQ. Pokud k tomu dojde, chybové zprávy se zapisují do protokolů chyb produktu WebSphere MQ a potenciálně může dojít k selhání volání MQBEGIN. Chcete-li pomoci zajistit, aby operační systém neselhal při načítání zaváděcího souboru přepínače, musíte splnit tyto požadavky:

1. Soubor načtení přepínače musí být k dispozici v umístění, které je zadáno v souboru qm.ini .
2. Soubor načtení přepínače musí být přístupný pro všechny procesy, které je třeba načíst, včetně procesů správce front a aplikačních procesů.
3. Všechny knihovny, na kterých soubor načtení přepínače závisí, včetně knihoven, které poskytuje databázový produkt, musí být přítomné a přístupné.

Přidání informací o konfiguraci do správce front
Pokud jste vytvořili zaváděcí soubor přepínače pro správce databází a umístili jej do bezpečného umístění, musíte toto umístění uvést do svého správce front.

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Chcete-li určit umístění, proveďte následující kroky:

- Na systémech Windows a Linux (platformyx86 a x86-64) použijte produkt WebSphere MQ Explorer. V panelu vlastností správce front určete podrobnosti o souboru načtení přepínače v panelu vlastností správce front.
- Ve všech ostatních systémech určete podrobnosti o souboru načtení přepínače ve stanze XAResourceManager v souboru qm.ini správce front.

Přidejte objekt stanza XAResourceManager pro databázi, kterou bude váš správce front koordinovat. Nejběžnějším případem je, že existuje pouze jedna databáze, a tedy pouze jedna stanza XAResourceManager . Podrobnosti o komplikovanější konfiguraci zahrnující více databází viz “Konfigurace více databází” na stránce 58. Atributy objektu stanza XAResourceManager jsou následující:

Název=název
Uživatelsky zvolený řetězec, který identifikuje správce prostředků. V důsledku toho bude pojmenujte objekt stanza XAResourceManager . Název je povinný a může mít délku až 31 znaků.

Název, který vyberete, musí být jedinečný; musí existovat pouze jedna stanza XAResourceManager s tímto názvem v tomto souboru qm.ini . Název by měl být také smysluplný, protože jej správce front používá při odkazování na tohoto správce prostředků ve zprávách protokolu chyb správce front a ve výstupu při použití příkazu `dspmqtrn` . (Další informace viz “Zobrazení neprovedených jednotek práce s příkazem dspmqtrn” na stránce 60 .)

Jakmile jste vybrali název a spustili správce front, neměňte atribut Název. Další informace o změně informací o konfiguraci najdete v tématu “Změna konfiguračních informací” na stránce 62.

SwitchFile= název
Jedná se o název souboru načtení přepínače XA, který jste vytvořili dříve. Toto je povinný atribut. Kód ve správci front a aplikační procesy produktu WebSphere MQ se pokusí načíst soubor pro načtení přepínače při dvou příležitostech:

1. Při spuštění správce front
2. Při prvním volání produktu MQBEGIN v aplikačním procesu produktu WebSphere MQ

Atributy zabezpečení a oprávnění vašeho souboru načtení přepínače musí těmto procesům povolit provedení této akce.

XAOpenString= řetězec
Jedná se o řetězec dat, který kód produktu WebSphere MQ předává ve svých voláních funkci xa_open správce databáze. Jedná se o volitelný atribut; pokud je vynechán, předpokládá se řetězec s nulovou délkou.

Kód ve správci front a aplikační procesy produktu WebSphere MQ volají funkci xa_open při dvou příležitostech:

1. Při spuštění správce front
2. Při prvním volání produktu MQBEGIN v aplikačním procesu produktu WebSphere MQ

Formát pro tento řetězec je specifický pro každý databázový produkt a bude popsán v dokumentaci k danému produktu. Řetězec xa_open obsahuje informace o ověření (jméno uživatele a heslo), které umožňuje povolit připojení k databázi jak ve správci front, tak i v aplikačních procesech.

XACloseString= řetězec
Jedná se o řetězec dat, který kód produktu WebSphere MQ předává ve svých voláních funkci xa_close správce databáze. Jedná se o volitelný atribut; pokud je vynechán, předpokládá se řetězec s nulovou délkou.

Kód ve správci front a aplikační procesy produktu WebSphere MQ volají funkci xa_close při dvou příležitostech:

1. Při spuštění správce front
2. Při volání MQDISC ve vašem procesu aplikace WebSphere MQ , které dříve učinilo volání MQBEGIN

Formát pro tento řetězec je specifický pro každý databázový produkt a bude popsán v dokumentaci k danému produktu. Obecně je řetězec prázdný a je běžné vynechat atribut XACloseString z objektu stanza XAResourceManager .

ThreadOfControl=THREAD |PROCESS

Řídicí hodnota ThreadOfmůže být THREAD nebo PROCESS. Správce front ji používá pro účely serializace. Jedná se o volitelný atribut; pokud je vynechán, předpokládá se hodnota PROCESS.

Pokud kód klienta databáze umožňuje podprocesy volání funkcí XA bez serializace, může být hodnota parametru ThreadOfControl THREAD. Správce front předpokládá, že v případě potřeby může volat funkce XA ve sdílené knihovně klienta databáze z více podprocesů současně.

Pokud kód klienta databáze nepovoluje podprocesy, aby volaly své funkce XA tímto způsobem, musí být hodnota atributu ThreadOfControl PROCESS. V takovém případě správce front provede serializaci všech volání do sdílené knihovny klienta databáze, takže v určitém procesu bude provedeno pouze jedno volání v daném okamžiku. Pravděpodobně byste také měli zajistit, aby vaše aplikace provedla podobnou serializaci, pokud je spuštěna s více podprocesy.

Všimněte si, že tento problém, který je schopen se s tímto způsobem vypořádat s procesy s podporou více procesů, je problémem pro dodavatele tohoto produktu. Podrobnosti o tom, zda můžete nastavit atribut ThreadOfControl na THREAD nebo PROCESS, naleznete v dokumentaci k databázovému produktu. Doporučujeme, abyste, pokud můžete, jste nastavili ThreadOfControl na THREAD. Pokud je na pochybách, volba *bezpečnější* ji nastaví na PROCESS, i když ztratíte potenciální výhody výkonu funkce THREAD.

Psaní a úprava vašich aplikací
Jak implementovat globální jednotku práce.

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Vzorové aplikační programy pro scénář 1 globální jednotky práce dodávané s instalací produktu WebSphere MQ jsou popsány v příručce “Představení pracovních jednotek” na stránce 40.

Obecně platí, že globální pracovní jednotka je implementována v aplikaci následující metodou (v pseudokódu):

```
MQBEGIN
MQGET
MQPUT
VLOŽENÍ SQL
MQCMIT
```

Účelem operace MQBEGIN je označení začátku globální transakce. Cílem MQCMIT je označení konce globální pracovní jednotky a dokončit ji se všemi zúčastněnými správci prostředků pomocí protokolu dvoufázového potvrzování.

Ve struktuře mezi MQBEGIN a MQCMIT správce front nevolá žádné volání do databáze, aby aktualizoval své prostředky. To znamená, že jediným způsobem, jak jsou tabulky databáze změněny, je váš kód (například SQL INSERT v pseudokódu).

Role správce front, pokud jde o databázi, je říct jí, kdy byla spuštěna globální jednotka práce, kdy skončila a zda by měla být globální transakce potvrzena nebo odvolána.

Pokud jde o vaši aplikaci, správce front provádí dvě role: správce prostředků (kde prostředky jsou zprávy ve frontách) a správce transakcí pro globální pracovní jednotku.

Začněte s dodávanými ukázkovými programy a pracujte s různými voláními rozhraní WebSphere MQ a rozhraním API databáze, které jsou v těchto programech prováděny. Dotyčná volání rozhraní API jsou plně zdokumentována v produktu “Ukázka programů WebSphere MQ” na stránce 92, Datové typy používané v modulu MQIa (v případě vlastního rozhraní API databáze) vlastní dokumentaci.

Testování systému
Víte, zda jsou vaše aplikace a systém správně konfigurovány pouze spuštěním těchto testů během testování. Můžete testovat konfiguraci systému (úspěšnou komunikaci mezi správcem front a databází) tím, že sestavíte a spustíte jeden z dodaných ukázkových programů.

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Konfigurace Db2

Informace o podpoře a konfiguraci produktu DB2 .

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Úrovně podporované úrovně Db2 jsou definovány na stránce Podrobné systémové požadavky produktuIBM WebSphere MQ .

Poznámka: 32bitové instance Db2 nejsou podporovány na platformách, kde je správce front 64bitový.

Proveďte následující akce:

1. Zkontrolujte nastavení proměnných prostředí.
2. Vytvořte zaváděcí soubor přepínače Db2 .
3. Přidejte informace o konfiguraci správce prostředků.
4. V případě potřeby změňte konfigurační parametry Db2 .

Přečtěte si tyto informace ve spojení s obecnými informacemi poskytnutnými v produktu “Konfigurace systému pro koordinaci databáze” na stránce 45.

Varování: Spustíte-li produkt `db2profile` na platformách UNIX and Linux , nastaví se proměnná prostředí LIBPATH a LD_LIBRARY_PATH. Tyto proměnné prostředí se doporučuje `unset` , viz příslušná příručka *Začínáme* .

Kontrola nastavení proměnných prostředí Db2

Ujistěte se, že proměnné prostředí Db2 jsou nastaveny pro procesy správce front ***stejně jako v*** vašich aplikačních procesech. Zejména je třeba vždy nastavit proměnnou prostředí DB2INSTANCE ***před*** spuštěním správce front. Proměnná prostředí DB2INSTANCE identifikuje instanci Db2 obsahující aktualizované databáze Db2 . Příklad:

- V systémech UNIX and Linux použijte:

```
export DB2INSTANCE=db2inst1
```

- Na systémech Windows použijte:

```
set DB2INSTANCE=DB2
```

V systému Windows s databází Db2 je třeba přidat uživatele MUSR_MQADMIN do skupiny DB2USERS , a umožnit tak spuštění správce front.

Vytvoření zaváděcího souboru přepínače Db2

Nejjednodušším způsobem, jak vytvořit zaváděcí soubor přepínače Db2 , je použití ukázkového souboru xaswit.mak, který produkt WebSphere MQ poskytuje pro sestavení souborů načtení přepínače pro celou řadu databázových produktů.

Na systémech Windows můžete najít soubor xaswit.mak v adresáři *MQ_INSTALLATION_PATH*`\tools\c\samples\xatm`. *MQ_INSTALLATION_PATH* Představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován. Chcete-li vytvořit zaváděcí soubor přepínače Db2 se systémem Microsoft Visual C + +, použijte následující příkaz:

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

Vygenerovaný přepínací soubor je umístěn v `c:\Program Files\IBM\WebSphere MQ\exits`.

Soubor xaswit.mak můžete najít v adresáři *MQ_INSTALLATION_PATH*`/samp/xatm`. *MQ_INSTALLATION_PATH* Představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

Upravte soubor xaswit.mak tak, aby *odkomentuje* řádky odpovídající verzi produktu Db2 , kterou používáte. Potom proveďte soubor Makefile pomocí příkazu:

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

Vygenerovaný 32bitový zaváděcí soubor se umístí do `/var/mqm/exits`.

Vygenerovaný 64bitový zaváděcí soubor přepínače je umístěn v `/var/mqm/exits64`.

Přidání informací o konfiguraci správce prostředků pro Db2

Chcete-li deklarovat Db2 jako účastníka globálních transakcí, musíte upravit informace o konfiguraci správce front. Úprava informací o konfiguraci tímto způsobem je popsána v dalších podrobnostech produktu “Přidání informací o konfiguraci do správce front” na stránce 46.

- V systémech Windows a Linux (platformyx86 a x86-64) použijte Průzkumníka produktu WebSphere MQ . V panelu vlastností správce front určete podrobnosti o souboru načtení přepínače v panelu vlastností správce front.
- Ve všech ostatních systémech určete podrobnosti o souboru načtení přepínače ve stanze XAResourceManager v souboru qm.ini správce front.

Obrázek 9 na stránce 50 je ukázka systému UNIX zobrazující položku XAResourceManager , kde má být databáze koordinována s názvem `mydbname`, přičemž tento název je zadán v modulu `XAOpenString`:

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

Obrázek 9. Ukázka položky XAResourceManager pro Db2 na platformách UNIX

Poznámka:

1. Produkt `ThreadOfControl=THREAD` nelze použít s verzí Db2 dřívějších než verze 8. Nastavte parametr `ThreadOfControl` a parametr XAOpenString `toc` na jednu z následujících kombinací:
   - `ThreadOfControl=THREAD` a `toc=t`
   - `ThreadOfControl=PROCESS` a `toc=p`

   Pokud používáte k povolení koordinace JDBC/JTA obslužný soubor přepínače jdbcdb2 XA, musíte použít `ThreadOfControl=PROCESS` a `toc=p`.

Změna konfiguračních parametrů Db2

Pro každou databázi Db2 , kterou správce front koordinuje, je třeba nastavit oprávnění k databázi, změnit parametr tp_mon_name a resetovat parametr maxappls. Chcete-li to provést, proveďte následující kroky:

Nastavení oprávnění k databázi

Procesy správce front jsou spouštěny s efektivním uživatelem a skupinou mqm na systémech UNIX and Linux . V systémech Windows se spouštějí jako uživatel, který spustil správce front. Může se jednat o jednu z následujících možností:

1. Uživatel, který vydal příkaz `strmqm` , nebo
2. Uživatel, pod kterým je spuštěna služba IBM MQSeries Service COM

Při výchozím nastavení se tento uživatel nazývá MUSR_MQADMIN.

Pokud jste v řetězci xa_open neuvedli jméno uživatele a heslo, použije Db2 k ověření volání xa_open **uživatel, pod kterým je spuštěn správce front** . Pokud tento uživatel (například uživatel mqm v systému UNIX and Linux) nemá v databázi minimální oprávnění, databáze odmítne ověřit volání xa_open.

Stejné úvahy platí pro váš aplikační proces. Pokud jste v řetězci xa_open neuvedli jméno uživatele a heslo, použije uživatel Db2 uživatele, pod kterým je vaše aplikace spuštěna, k ověření volání xa_open, které bylo provedeno během prvního volání MQBEGIN. I tento uživatel musí mít v databázi minimální oprávnění, aby to fungovalo.

Například dejte uživateli mqm oprávnění k připojení v databázi mydbname zadáním následujících příkazů Db2 :

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

Další informace o zabezpečení viz “Doporučení ohledně zabezpečení” na stránce 59 .

Windows **Změňte parametr TP_MON_NAME**

Pouze v případě systému Db2 pro systémy Okna změňte konfigurační parametr TP_MON_NAME tak, aby pojmenuje knihovnu DLL, kterou produkt Db2 používá k volání správce front pro dynamickou registraci.

Použijte příkaz `db2 update dbm cfg using TP_MON_NAME mqmax` na název MQMAX.DLL jako knihovna, kterou produkt Db2 používá k volání správce front. Musí být k dispozici v adresáři v rámci proměnné PATH.

Reset parametru maxappls

Možná budete muset zkontrolovat nastavení pro parametr *maxappls* , který omezuje maximální počet aplikací, které mohou být připojeny k databázi. Další informace naleznete v části “Instalace a konfigurace databázového produktu” na stránce 45.

Konfigurace Oracle

Informace o podpoře a konfiguraci Oracle .

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Postupujte takto:

1. Zkontrolujte nastavení proměnných prostředí.
2. Vytvořte zaváděcí soubor přepínače Oracle .
3. Přidejte informace o konfiguraci správce prostředků.
4. V případě potřeby změňte konfigurační parametry Oracle .

Aktuální seznam úrovní produktu Oracle podporovaných produktem IBM WebSphere MQ je k dispozici na stránce Podrobné systémové požadavky produktuIBM WebSphere MQ .

Kontrola nastavení proměnných prostředí Oracle

Ujistěte se, že jsou proměnné prostředí Oracle nastaveny pro procesy správců front stejně jako v aplikačních procesech. Před spuštěním správce front je vždy třeba nastavit následující proměnné prostředí:

DOMOVSKÝ_ADRESÁŘ_ORACLE_HOME
Domovský adresář Oracle . Např. na systémech UNIX and Linux použijte:

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

V systémech Windows použijte:

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

ORACLE_SID
Používá se SID Oracle . Pokud používáte Net8 pro připojitelnost klienta/serveru, možná nemusíte tuto proměnnou prostředí nastavit. Konzultujte dokumentaci Oracle .

Následný příklad je příkladem nastavení této proměnné prostředí na systémech UNIX and Linux :

```
export ORACLE_SID=sid1
```

Ekvivalentem na systémech Windows je:

```
set ORACLE_SID=sid1
```

Poznámka: Proměnná prostředí PATH musí být nastavena tak, aby obsahovala adresář binárních souborů (například `ORACLE_INSTALL_DIR/VERSION/32BIT_NAME/bin` nebo `ORACLE_INSTALL_DIR/VERSION/64BIT_NAME/bin`), jinak se může zobrazit zpráva s informací o tom, že v počítači chybí knihovny oraclient.

Pokud provozujete správce front v 64bitových systémech Windows , musí být nainstalovány 64bitové i 32bitové klienty Oracle . Musíte nainstalovat oba klienty, protože správce front běží jako 32bitové procesy, které používají 32bitový zaváděcí soubor přepínače, který musí postupně spustit 32bitovou knihovnu DLL klienta Oracle .

Soubor načtení přepínače, načtený 64bitovými správci front, musí přistupovat ke knihovnám 64bitového klienta Oracle . 32bitoví správci front musí přistupovat ke 32bitovému klientu Oracle , je-li produkt IBM WebSphere MQ spuštěn na 64bitovém systému Windows .

Vytvoření zaváděcího souboru přepínače Oracle

Chcete-li vytvořit zaváděcí soubor přepínače Oracle , použijte ukázkový soubor `xaswit.mak`, který produkt IBM WebSphere MQ poskytuje k sestavení souborů pro načtení přepínače pro různé databázové produkty. Na systémech Windows můžete najít `xaswit.mak` v adresáři `C:\Program Files\IBM\WebSphere MQ\tools\c\samples\xatm`. Chcete-li vytvořit zaváděcí soubor přepínače Oracle s vizuálním produktem Microsoft Visual C + +, použijte: `nmake /f xaswit.mak oraswit.dll`

Vygenerovaný soubor přepínače je umístěn v *MQ_INSTALLATION_PATH*`\exits.`*MQ_INSTALLATION_PATH* Představuje vysokoúrovňový adresář, ve kterém je nainstalován produkt IBM WebSphere MQ .

`xaswit.mak` můžete najít v adresáři *MQ_INSTALLATION_PATH*`/samp/xatm.` *MQ_INSTALLATION_PATH* Představuje vysokoúrovňový adresář, ve kterém je nainstalován produkt IBM WebSphere MQ .

Upravte `xaswit.mak` a zrušte komentář u řádků odpovídajících verzi produktu Oracle , kterou používáte. Potom proveďte soubor Makefile pomocí příkazu:

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

Vygenerovaný 32bitový soubor pro načtení přepínače je umístěn v `/var/mqm/exits`.

Vygenerovaný 64bitový zaváděcí soubor přepínače je umístěn v `/var/mqm/exits64`.

Přidání konfiguračních informací správce prostředků pro Oracle

Musíte upravit informace o konfiguraci správce front tak, aby deklaroval Oracle jako účastníka v globálních pracovních jednotkách. Úprava informací o konfiguraci pro správce front tímto způsobem je podrobněji popsána v tématu “Přidání informací o konfiguraci do správce front” na stránce 46.

- Na systémech Windows a Linux (platformyx86 a x86-64) použijte produkt IBM WebSphere MQ Explorer. V panelu vlastností správce front určete podrobnosti o souboru načtení přepínače v panelu vlastností správce front.
- Ve všech ostatních systémech určete podrobnosti o souboru načtení přepínače ve stanze XAResourceManager v souboru `qm.ini` správce front.

Obrázek 10 na stránce 53 je ukázka systému UNIX and Linux zobrazující položku XAResourceManager . Do otevřeného řetězce XA musíte přidat `LogDir` , aby všechny informace o chybách a trasování byly zaprotokolovány do stejného místa.

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

Obrázek 10. Ukázková položka XAResourceManager pro Oracle na platformách UNIX and Linux

Poznámka:

1. V produktu Obrázek 10 na stránce 53byl řetězec xa_open použit se čtyřmi parametry. Další parametry lze zahrnout podle popisu v dokumentaci produktu Oracle.
2. Používáte-li parametr IBM WebSphere MQ `ThreadOfControl=THREAD` , musíte použít parametr Oracle `+threads=true` ve stanze XAResourceManager .

Další informace o řetězci xa_open naleznete v příručce *Oracle8 Server Application Developer's Guide* .

Změna konfiguračních parametrů Oracle

Pro každou databázi Oracle , kterou správce front koordinuje, je třeba zkontrolovat maximální počet relací a nastavit oprávnění k databázi. Chcete-li to provést, postupujte takto:

Přezkoumat maximální počet relací
Možná budete muset zkontrolovat nastavení LICENSE_MAX_SESSIONS a PROCESSES, abyste mohli vzít v úvahu další připojení vyžadovaná procesy náležícími ke správci front. Další podrobnosti naleznete v části “Instalace a konfigurace databázového produktu” na stránce 45.

Nastavení oprávnění k databázi
Jméno uživatele Oracle uvedené v řetězci xa_open musí mít oprávnění pro přístup k pohledu DBA_PENDING_TRANSACTIONS, jak je popsáno v dokumentaci Oracle .

Potřebné oprávnění lze udělit pomocí následujícího příkazu:

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

Konfigurace modulu Informix

Informace o podpoře a konfiguraci systému Informix .

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Postupujte takto:

1. Ujistěte se, že jste nainstalovali odpovídající sadu SDK klienta Informix :

- 32bitoví správci front a aplikace vyžadují 32bitovou sadu SDK klienta Informix .
- 64bitoví správci front a aplikace vyžadují 64bitovou sadu SDK klienta Informix .

2. Ujistěte se, že databáze Informix jsou vytvořeny správně.
3. Zkontrolujte nastavení proměnných prostředí.
4. Sestavte zaváděcí soubor přepínače Informix .
5. Přidejte informace o konfiguraci správce prostředků.

Aktuální seznam úrovní produktu Informix podporovaných produktem WebSphere MQ je k dispozici na stráncePodrobné systémové požadavky produktuIBM WebSphere MQ .

Ujištění, že jsou databáze Informix vytvořeny správně

Každá databáze Informix , která má být koordinována správcem front produktu WebSphere MQ , musí být vytvořena zadáním parametru `log` . Příklad:

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

Správci front produktu WebSphere MQ nejsou schopni koordinovat databáze Informix , které nemají parametr `log` určený při vytváření. Pokud se správce front pokusí koordinovat databázi Informix , která nemá zadán parametr `log` při vytvoření, volání xa_open na Informix selže a vygeneruje se počet chyb FFST .

Kontrola nastavení proměnných prostředí Informix

Ujistěte se, že jsou proměnné prostředí Informix nastaveny pro procesy správce front ***stejně jako v*** vašich aplikačních procesech. Zejména vždy nastavte následující proměnné prostředí **před** spuštěním správce front:

INFORMIXDIR
Adresář instalace produktu Informix .

- Pro 32 bitové aplikace UNIX and Linux použijte tento příkaz:

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

- Pro 64bitové aplikace UNIX and Linux použijte tento příkaz:

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

- Pro aplikace Windows použijte tento příkaz:

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

Pro systémy s 64bitovými správci front, kteří musí podporovat 32bitové i 64bitové aplikace, je třeba nainstalovat obě 32bitové a 64bitové sady SDK klienta Informix . Ukázkový soubor Makefile `xaswit.mak`použitý pro vytvoření souboru načtení přepínače také nastaví instalační adresáře produktu.

INFORMAČNÍ SERVER
Název serveru Informix . Např. na systémech UNIX and Linux použijte:

```
export INFORMIXSERVER=hostname_1
```

Na systémech Windows použijte:

```
set INFORMIXSERVER=hostname_1
```

ONCONFIG
Název konfiguračního souboru serveru Informix . Např. na systémech UNIX and Linux použijte:

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

Na systémech Windows použijte:

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

Vytvoření zaváděcího souboru přepínače Informix

Chcete-li vytvořit soubor načtení přepínače Informix , použijte ukázkový soubor xaswit.mak, který produkt WebSphere MQ poskytuje pro sestavení souborů načtení přepínače pro různé databázové produkty. Na systémech Windows můžete najít soubor xaswit.mak v adresáři *MQ_INSTALLATION_PATH*\tools\c\samples\xatm. *MQ_INSTALLATION_PATH* Představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován. Chcete-li vytvořit zaváděcí soubor přepínače Informix se systémem Microsoft Visual C + +, použijte následující příkaz:

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

Vygenerovaný přepínací soubor je umístěn v `c:\Program Files\IBM\WebSphere MQ\exits`.

Soubor xaswit.mak můžete najít v adresáři *MQ_INSTALLATION_PATH*/samp/xatm. *MQ_INSTALLATION_PATH* Představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

Upravte soubor xaswit.mak tak, aby *odkomentuje* řádky odpovídající verzi produktu Informix , kterou používáte. Potom proveďte soubor Makefile pomocí příkazu:

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

Vygenerovaný 32bitový soubor pro načtení přepínače je umístěn v `/var/mqm/exits`.

Vygenerovaný 64bitový zaváděcí soubor přepínače je umístěn v `/var/mqm/exits64`.

Přidání informací o konfiguraci správce prostředků pro produkt Informix

Chcete-li deklarovat produkt Informix jako účastníka globálních transakcí, musíte upravit informace o konfiguraci správce front. Úprava informací o konfiguraci pro správce front tímto způsobem je podrobněji popsána v tématu “Přidání informací o konfiguraci do správce front” na stránce 46.

- V systémech Windows a Linux (platformyx86 a x86-64) použijte Průzkumníka produktu WebSphere MQ . V panelu vlastností správce front určete podrobnosti o souboru načtení přepínače v panelu vlastností správce front.
- Ve všech ostatních systémech určete podrobnosti o souboru načtení přepínače ve stanze XAResourceManager v souboru qm.ini správce front.

Obrázek 11 na stránce 56 je ukázka systému UNIX zobrazující položku qm.ini XAResourceManager , kde má být databáze koordinována s názvem `mydbname`, přičemž tento název je zadán v souboru `XAOpenString`:

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

Obrázek 11. Ukázka položky XAResourceManager pro Informix na platformách UNIX

Poznámka: Ukázka xaswit.mak na platformách UNIX standardně vytváří soubor zaváděcího zdroje, který používá knihovny Informix s podporou podprocesů. Při použití těchto knihoven produktu Informix je třeba zkontrolovat, zda je ovládací prvek ThreadOfnastaven na hodnotu THREAD. V produktu Obrázek 11 na stránce 56je atribut objektu stanza qm.ini XAResourceManager ThreadOfControl nastaven na THREAD. Je-li uvedeno THREAD, aplikace musí být sestaveny pomocí vláken Informix se závitem a knihoven API vláken WebSphere MQ .

Atribut XAOpenString musí obsahovat název databáze, za nímž následuje symbol @, a za ním následuje název serveru Informix .

Chcete-li používat nevláknové knihovny Informix , musíte se ujistit, že atribut objektu stanza qm.ini XAResourceManager ThreadOfControl je nastaven na PROCESS. Musíte také provést následující změny v ukázce xaswit.mak:

1. Odkomentujte generování nevláknového zaváděcího souboru přepínače.
2. Označte jako komentář generování zaváděcího souboru s podporou podprocesů se závitem.

Sybase Konfigurace

Informace o podpoře a konfiguraci Sybase .

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Postupujte takto:

1. Ujistěte se, že jste nainstalovali knihovny Sybase XA, například instalací volby XA DTM.
2. Zkontrolujte nastavení proměnných prostředí.
3. Povolte podporu standardu Sybase XA.
4. Vytvořte zaváděcí soubor přepínače Sybase .
5. Přidejte informace o konfiguraci správce prostředků.

Aktuální seznam úrovní produktu Sybase podporovaných produktem WebSphere MQ je k dispozici na stránce Podrobné systémové požadavky produktuIBM WebSphere MQ .

Kontrola nastavení proměnných prostředí Sybase

Ujistěte se, že jsou proměnné prostředí Sybase nastaveny pro procesy správce front ***stejně jako v*** vašich aplikačních procesech. Zejména vždy nastavte následující proměnné prostředí **před** spuštěním správce front:

Sybase
Umístění instalace produktu Sybase . Např. na systémech UNIX and Linux použijte:

```
export SYBASE=/sybase
```

Na systémech Windows použijte:

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

SYBASE_OCS
Adresář pod SYBASE, kam jste nainstalovali klientské soubory Sybase . Např. na systémech UNIX and Linux použijte:

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

Na systémech Windows použijte:

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

Povolení podpory standardu Sybase XA

V konfiguračním souboru Sybase XA $SYBASE/$SYBASE_OCS/xa_configdefinujte logický Resource Manager (LRM) pro každé připojení k serveru Sybase , který se aktualizuje. Příklad obsahu souboru $SYBASE/$SYBASE_OCS/xa_config je uveden v části Obrázek 12 na stránce 57.

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

Obrázek 12. Příklad obsahu $SYBASE/ $SYBASE_OCS/xa_config

Vytvoření zaváděcího souboru přepínače Sybase

Chcete-li vytvořit zaváděcí soubor přepínače Sybase , použijte ukázkové soubory dodávané s produktem WebSphere MQ.Na systémech Windows můžete najít soubor xaswit.mak v adresáři C:\Program Files\IBM\WebSphere MQ\tools\c\samples\xatm. Chcete-li vytvořit zaváděcí soubor přepínače Sybase s produktem Microsoft Visual C + +, použijte následující příkaz:

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

Vygenerovaný přepínací soubor je umístěn v c:\Program Files\IBM\WebSphere MQ\exits.

Soubor xaswit.mak můžete najít v adresáři *MQ_INSTALLATION_PATH*/samp/xatm. *MQ_INSTALLATION_PATH* Představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

Upravte soubor xaswit.mak tak, aby *odkomentuje* řádky odpovídající verzi produktu Sybase , kterou používáte. Potom proveďte soubor Makefile pomocí příkazu:

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

Vygenerovaný 32bitový zaváděcí soubor se umístí do /var/mqm/exits.

Vygenerovaný 64bitový zaváděcí soubor přepínače je umístěn v /var/mqm/exits64.

Přidání informací o konfiguraci správce prostředků pro Sybase

Musíte upravit informace o konfiguraci správce front tak, aby deklaroval Sybase jako účastníka v globálních jednotkách práce. Úprava informací o konfiguraci je podrobněji popsána v tématu “Přidání informací o konfiguraci do správce front” na stránce 46.

- V systémech Windows a Linux (platformyx86 a x86-64) použijte Průzkumníka produktu WebSphere MQ . V panelu vlastností správce front určete podrobnosti o souboru načtení přepínače v panelu vlastností správce front.
- Ve všech ostatních systémech určete podrobnosti o souboru načtení přepínače ve stanze XAResourceManager v souboru qm.ini správce front.

Obrázek 13 na stránce 58 zobrazuje ukázku UNIX and Linux , která používá databázi přidruženou k definici LRM s názvem *lrmname* v konfiguračním souboru Sybase XA $SYBASE/$SYBASE_OCS/xa_config. Chcete-li protokolovat volání funkcí XA do protokolu, zahrňte název souboru protokolu:

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

Obrázek 13. Ukázka položky XAResourceManager pro Sybase na platformách UNIX and Linux

Použití vícevláknových programů s Sybase

Pokud používáte vícevláknové programy s globálními jednotkami WebSphere MQ , které obsahují aktualizace do Sybase, ***musíte*** použít hodnotu THREAD pro parametr Řízení ThreadOf. Také se ujistěte, že propojíte váš program (a zaváděcí soubor přepínače) se zajištěním neporušenosti vláken Sybase libraries (verze _r). Použití hodnoty THREAD pro parametr Řízení ThreadOfje zobrazeno v Obrázek 13 na stránce 58.

Konfigurace více databází

Chcete-li správce front nakonfigurovat tak, aby aktualizace na více databází mohly být zahrnuty do globálních pracovních jednotek, přidejte objekt stanza XAResourceManager pro každou databázi.

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Pokud jsou databáze všechny spravovány stejným správcem databází, každá sekce definuje samostatnou databázi. Každá stanza uvádí stejné *SwitchFile*, ale obsah *XAOpenString* se liší, protože uvádí název aktualizované databáze. Například stanzy zobrazené v produktu Obrázek 14 na stránce 58 konfigurují správce front s databází Db2 *MQBankDB* a *MQFeeDB* na systémech UNIX and Linux .

Důležité: Nemůžete mít více sekcí odkazujících na stejnou databázi. Tato konfigurace nefunguje za žádných okolností, a pokud se pokusíte tuto konfiguraci, dojde k selhání.

Obdržíte chyby ve formuláři `when the MQ code makes its second xa_open call in any process in this environment, the database software fails the second xa_open with a -5 error, XAER_INVAL`.

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

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

Obrázek 14. Ukázka položek XAResourceManager pro více databází Db2

Pokud jsou databáze, které mají být aktualizovány, spravovány různými správci databází, přidejte pro každou z nich sekci XAResourceManager . V tomto případě každá stanza uvádí jiný *SwitchFile*. Je-li například produkt *MQFeeDB* spravován systémem Oracle namísto DB2, použijte následující oddíly v systémech UNIX and Linux :

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

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

Obrázek 15. Ukázka položek XAResourceManager pro databázi DB2 a Oracle

V zásadě neexistuje žádné omezení počtu instancí databáze, které lze nakonfigurovat pomocí jednoho správce front.

Poznámka: Informace o podpoře pro zahrnutí databází Informix do více aktualizací databáze v rámci globálních pracovních jednotek naleznete v souboru Readme produktu.

Doporučení ohledně zabezpečení

Pokyny pro spuštění vaší databáze v rámci modelu XA.

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Následující informace jsou poskytnuty pouze pro navádění. Ve všech případech se podívejte do dokumentace dodané se správcem databází, abyste určili důsledky zabezpečení provozu vaší databáze pod modelem XA.

Proces aplikace označuje začátek globální jednotky práce pomocí příkazového slova `MQBEGIN` . První volání `MQBEGIN` , které se připojuje k problémům aplikace, se připojuje ke všem zúčastněným databázím voláním jejich kódu knihovny klienta v vstupním bodu xa_open. Všichni správci databází poskytují mechanizmus pro zadávání ID uživatele a hesla ve svém XAOpenString. To je jediná doba, kdy autentizační informace proudí.

Všimněte si, že na platformách UNIX and Linux musí být aplikace rychlé cesty spuštěny s efektivním ID uživatele mqm při provádění volání MQI.

Aspekty při ztrátě kontaktu se správcem prostředků XA

Správce front toleruje správce databází, kteří nejsou dostupní. To znamená, že můžete správce front spustit a zastavit nezávisle na databázovém serveru. Po obnovení kontaktu se správce front a databáze znovu synchronizují. Můžete také použít příkaz rsvmqtrn k ručnímu vyřešení transakcí v nejistém stavu.

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Při běžném provozu je nutné po dokončení kroků konfigurace pouze minimální množství administrace. Administrativní úloha je usnadněna, protože správce front toleruje správce databází, kteří nejsou k dispozici. To znamená zejména, že:

- Správce front může být spuštěn kdykoli, aniž by nejprve každý správce databází spouštěl všechny správce databází.
- Správce front se nemusí zastavit a restartovat, pokud jeden ze správců databází přestane být k dispozici.

To vám umožní spustit a zastavit správce front nezávisle na databázovém serveru.

Dojde-li ke ztrátě kontaktu mezi správcem front a databází, je třeba je znovu synchronizovat, je-li k dispozici znovu. Resynchronizace je proces, při kterém jsou dokončeny všechny neověřené jednotky práce zahrnující danou databázi. Obecně platí, že k tomu dojde automaticky, aniž by byla nutná intervence uživatele. Správce front požádá databázi o seznam jednotek práce, které mají pochybnosti. Poté instruuje databázi buď potvrdit, nebo odvolat každou z těchto sporných jednotek práce.

Když se správce front spustí, resynchronizuje se s každou databází. Když se individuální databáze stane nedostupnou, je třeba znovu synchronizovat pouze tuto databázi, pokud správce front zjistí, že je opět k dispozici.

Správce front znovu získá kontakt s dříve nedostupnou databází, protože nové globální jednotky práce jsou spuštěny s produktem MQBEGIN. To znamená, že volá funkci xa_open v knihovně klienta databáze. Pokud tento volání xa_open selže, funkce MQBEGIN se vrátí s kódem dokončení MQCC_WARNING a kódem příčiny MQRC_PARTICanANT_NOT_AVAILABLE. Volání MQBEGIN můžete zopakovat později.

Nepokračujte v pokusu o globální pracovní jednotku, která zahrnuje aktualizace databáze, která selhala během operace MQBEGIN. K této databázi nebude existovat připojení, přes které lze provést aktualizace. Jediným možností je ukončit program nebo zopakovat spuštění funkce MQBEGIN v naději, že se databáze může znovu zpřístupnit.

Případně můžete použít příkaz `rsvmqtrn` k explicitnímu vyřešení všech nejistých transakcí.

Nejisté jednotky práce

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Databáze může být ponechána nepochybným jednotkám práce, pokud je kontakt se správcem front ztracen poté, co byl správce databází instruován k přípravě. Do okamžiku, kdy databázový server obdrží výsledek od správce front (potvrdit nebo odvolat), je nutné zachovat zámky databáze přidružené k aktualizacím.

Vzhledem k tomu, že tyto zámky brání jiným aplikacím v aktualizaci nebo čtení databázových záznamů, je třeba provést opětovnou synchronizaci co nejdříve.

Pokud z nějakého důvodu nemůžete čekat na automatickou synchronizaci správce front s databází, můžete použít nástroje poskytované správcem databáze k ručnímu potvrzení nebo odvolání aktualizací databáze. Ve specifikaci *X/Open Distributed Transaction Processing: Specifikace XA*se nazývají rozhodnutí *heuristické* . Použijte jej pouze jako poslední možnost, protože hrozí ohrožování integrity dat; můžete například omylem odvolat aktualizace databáze, když všechny ostatní účastníci potvrdí své aktualizace.

Je mnohem lepší restartovat správce front nebo použít příkaz `rsvmqtrn` , když byla databáze restartována, aby se zahájila automatická resynchronizace.

Zobrazení neprovedených jednotek práce s příkazem dspmqtrn

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Když je správce databází nedostupný, můžete použít příkaz **dspmqtrn** ke kontrole stavu nevyřízených globálních jednotek práce zahrnující tuto databázi.

Příkaz **dspmqtrn** zobrazí pouze ty jednotky práce, ve kterých je jeden nebo více účastníků nejistých. Účastníci čekají na rozhodnutí od správce front, aby potvrdili nebo odvolali připravené aktualizace.

Pro každou z těchto globálních jednotek práce se stav každého účastníka zobrazí ve výstupu příkazu **dspmqtrn**. Pokud jednotka práce neaktualizovala prostředky konkrétního správce prostředků, nezobrazuje se.

Pokud jde o neověřující pracovní jednotku, znamená to, že správce prostředků provedl jednu z následujících možností:

Připraveno
Správce prostředků je připraven potvrdit své aktualizace.

Potvrzené
Správce prostředků potvrdil své aktualizace.

Válcované
Správce prostředků odvolal své aktualizace.

Zúčastněný
Správce prostředků je účastník, ale nebyl připraven, potvrzen nebo odvolán jeho aktualizace.

Po restartování správce front zeptá každá databáze mající objekt stanza XAResourceManager pro seznam svých nejistých globálních pracovních jednotek. Pokud databáze nebyla restartována nebo pokud je jinak nedostupná, správce front nemůže ještě dodat do databáze konečné výsledky pro tyto jednotky práce. Výsledek sporných jednotek práce se doručí do databáze při první příležitosti, kdy je databáze opět k dispozici.

V takovém případě je správce databází hlášen jako stav v *připraveném* stavu, dokud se nesynchronizuje resynchronizace.

Kdykoli příkaz `dspmqtrn` zobrazí spornou transakci, bude nejprve uveden seznam všech možných správců prostředků, kteří se mohou podílet. Jedná se o alokovaný jedinečný identifikátor *RMId*, který se používá namísto *Názvu* správců prostředků při ohlášení stavu s ohledem na neověřovanou jednotku práce.

Výstup příkazu dspmqtrn zobrazuje výsledek zadání následujícího příkazu:

```
dspmqtrn -m MY_QMGR
```

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

AMQ7056: Transaction number 0,1.
    XID: formatID 5067085, gtrid_length 12, bqual_length 4
         gtrid [3291A50600002013746573 74]
         bqual [00000001]
AMQ7105: Resource manager 0 has committed.
AMQ7104: Resource manager 1 has prepared.
AMQ7104: Resource manager 2 has prepared.
```

kde *Číslo transakce* je ID transakce, kterou lze použít s příkazem rsvmqtrn . Další informace o zprávě AMQ7056 naleznete v příručce AMQ7000-7999: WebSphere MQ . Proměnné *XID* jsou součástí specifikace *X/Open XA Specification*; pro většinu aktuální informace o této specifikaci naleznete následující informace: https://publications.opengroup.org/c193.

Obrázek 16. Ukázka výstupu dspmqtrn

Výstup ve výstupu Ukázka výstupu dspmqtrn ukazuje, že ke správci front jsou přidruženi tři správci prostředků. Prvním z nich je správce prostředků `0`, který je sám správcem front. Další dvě instance správce prostředků jsou databáze `MQBankDB` a `MQFeeDB` Db2 .

Tento příklad ukazuje pouze jedinou neověřnou jednotku práce. Je vydána zpráva pro všechny tři správce prostředků, což znamená, že byly provedeny aktualizace správce front a obou databází Db2 v rámci pracovní jednotky.

Aktualizace provedené ve správci front, správci prostředků **0**, byly *potvrzeny*. Aktualizace databází Db2 se nacházejí ve stavu *Připraveno* , což znamená, že produkt Db2 se musí stát nedostupným, než bude volán k potvrzení aktualizací databází *MQBankDB* a *MQFeeDB* .

Pochybná jednotka práce má externí identifikátor nazvaný XID (*transaction id*). Jedná se o součást dat, která je dána správci front produktu Db2 za účelem identifikace její části globální jednotky práce.

Řešení neprovedených jednotek práce s příkazem rsvmqtrn
Neprovedené jednotky práce jsou dokončeny, když je správce front a DB2 znovu synchronizováno.

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Výstup zobrazený v souboru Obrázek 16 na stránce 61 představuje jednu neověřenou jednotku práce, ve které dosud nebylo doručeno rozhodnutí o potvrzení pro databáze DB2 .

Chcete-li dokončit tuto pracovní jednotku, musí být správce front a produkt DB2 znovu synchronizovány, je-li k dispozici další položka DB2 . Správce front používá nové jednotky práce jako příležitost k opětovnému získání kontaktu s produktem DB2. Případně můžete správci front nařídit, aby resynchronizovat explicitně pomocí příkazu **rsvmqtrn** .

Po restartování produktu DB2 proveďte restart všech zámků databáze, které jsou přidruženy k neověřované jednotce práce, a které jsou uvolněny co nejrychleji. Použijte volbu -a, která říká správci front, aby vyřešil všechny neověřené transakce. V následujícím příkladu byl produkt DB2 restartován, takže správce front může vyřešit nejistou jednotku práce:

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

Smíšené výsledky a chyby
Ačkoli správce front používá protokol s dvoufázovým potvrzováním, není zcela odstraněn možnost některých jednotek práce, které se dokončují smíšenými výsledky. To znamená, že některé účastníky potvrdí své aktualizace a některé jejich aktualizace se vrátí zpět.

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Jednotky práce, které jsou dokončené se smíšeným výsledkem, mají závažný dopad, protože sdílené prostředky, které měly být aktualizovány jako jedna jednotka práce, již nejsou v konzistentním stavu.

Smíšené výsledky jsou způsobeny hlavně při heuristickém rozhodování o jednotkách práce namísto toho, aby správci front umožnil vyřešit vlastní neověřené jednotky práce. Taková rozhodnutí jsou mimo kontrolu správce front.

Kdykoli správce front zjistí smíšený výsledek, vytvoří informace FFST a dokumentuje selhání ve svých protokolech chyb, přičemž jedna ze dvou zpráv:

- Pokud se správce databáze odvolá, místo potvrzení:

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

- Pokud správce databáze potvrdí místo odvolání transakce, postupujte takto:

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

Další zprávy identifikují databáze, které jsou heuristicky poškozené. Pak je vaše odpovědnost za místní obnovení konzistence s ovlivněnou databází. Jedná se o komplikovaný proces, ve kterém musíte nejprve izolovat aktualizaci, která byla chybně potvrzena nebo odvolána, a vrátit se zpět nebo znovu provést změnu databáze ručně.

Změna konfiguračních informací
Poté, co správce front úspěšně zahájí koordinaci globálních pracovních jednotek, neměňte žádné informace o konfiguraci správce prostředků.

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Potřebujete-li změnit informace o konfiguraci, můžete to provést kdykoli, ale změny se neprojeví, dokud správce front nerestartujete.

Odeberete-li informace o konfiguraci správce prostředků pro databázi, efektivně odebíráte možnost správce front kontaktovat tohoto správce databází.

Nikdy změňte atribut *Název* v žádné z informací o konfiguraci správce prostředků. Tento atribut jedinečně identifikuje danou instanci správce databází pro správce front. Pokud změníte tento jedinečný identifikátor, bude správce front předpokládat, že databáze byla odebrána a že byla přidána zcela nová instance. Správce front stále přidružuje nevyřízené jednotky práce se starým názvem *Název*, pravděpodobně opouštějící databázi v nejistém stavu.

Odebrání instancí správce databází
Potřebujete-li odstranit databázi z konfigurace trvale, ujistěte se, že databáze není na pochybách, než správce front restartujete.

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Databázové produkty poskytují příkazy pro výpis neověřených transakcí. Pokud existují nějaké neověřené transakce, nejprve umožněte správci front, aby se resynchronizuje s databází. Proveďte to spuštěním správce front. Můžete ověřit, že byla provedena resynchronizace pomocí příkazu **rsvmqtrn** nebo vlastního příkazu databáze pro zobrazení sporných jednotek práce. Jakmile se ujistíte, že byla provedena resynchronizace, ukončete správce front a odeberte informace o konfiguraci databáze.

Pokud tento postup nedodržíte, správce front si stále pamatuje všechny neověřené jednotky práce zahrnující tuto databázi. Varovná zpráva AMQ7623se vydá pokaždé, když je správce front restartován. Pokud již tuto databázi s tímto správcem front nikdy nenakonfigurujete, použijte volbu -r příkazu **rsvmqtrn** , která dá pokyn správci front, aby zapomněl na účast databáze ve sporných transakcích. Správce front na takové transakce zapomene pouze v případě, že byly dokončeny neověřené transakce se všemi účastníky.

Existují okamžiky, kdy může být nutné dočasně odebrat některé informace o konfiguraci správce prostředků. Na systémech UNIX and Linux je toho nejlépe dosaženo tím, že odkomentujete stanzu, aby bylo možné jej později snadno obnovit. Pokud dojde k chybám při každém kontaktu správce front s konkrétní databází nebo správcem databází, můžete se rozhodnout, zda se tyto chyby vyskytnou. Dočasné odebrání příslušných informací o konfiguraci správce prostředků umožňuje správci front spustit globální jednotky práce zahrnující všechny ostatní účastníky. Zde je uveden příklad sekce s komentářem `XAResourceManager` :

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

Obrázek 17. Oddíl s komentářem XAResourceManager na systémech UNIX and Linux

V systémech Windows můžete pomocí Průzkumníka produktu WebSphere MQ odstranit informace o instanci správce databází. Při opětovném uvedení do původního stavu věnujte velkou pozornost zadávání správného názvu do pole *Název* . Pokud název chybně zadáte, můžete čelit problémům s nejistým stavem, jak je popsáno v tématu “Změna konfiguračních informací” na stránce 62.

Dynamická registrace XA

Specifikace XA poskytuje způsob snížení počtu volání `xa_*` , které správce transakcí provádí ve správci prostředků. Tato optimalizace je známá jako *dynamická registrace*.

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Dynamická registrace je podporována produktem DB2. Ostatní databáze jej mohou podporovat; podrobné informace naleznete v dokumentaci k databázovému produktu.

Proč je optimalizace dynamické registrace užitečná? Ve vaší aplikaci mohou některé globální jednotky práce obsahovat aktualizace databázových tabulek; jiné nemusí takové aktualizace obsahovat. Není-li v tabulkách databáze provedena žádná trvalá aktualizace, není třeba tuto databázi zahrnout do protokolu potvrzení, který se vyskytuje během operace MQCMIT.

Zda vaše databáze podporuje dynamickou registraci či nikoli, volání aplikace volá produkt `xa_open` během prvního volání MQBEGIN na připojení WebSphere MQ . Nazývá to také `xa_close` při následném volání MQDISC. Vzorek následných volání XA závisí na tom, zda databáze podporuje dynamickou registraci:

Pokud vaše databáze nepodporuje dynamickou registraci ...

Každá globální transakce zahrnuje několik volání funkce XA provedených kódem WebSphere MQ do knihovny klienta databáze bez ohledu na to, zda jste provedli trvalou aktualizaci tabulek této databáze v rámci vaší pracovní jednotky. Patří k nim:

- `xa_start` a `xa_end` z aplikačního procesu. Ty se používají k deklarování začátku a konce globální jednotky práce.
- `xa_prepare`, `xa_commit`a `xa_rollback` z procesu agenta správce front amqzlaa0. Ty se používají k dodání výsledku globální pracovní jednotky: rozhodnutí o potvrzení nebo odvolání transakce.

Kromě toho proces agenta správce front také volá produkt `xa_open` během první operace MQBEGIN.

Pokud vaše databáze podporuje dynamickou registraci ...

Kód WebSphere MQ provede pouze volání funkce XA, která jsou nezbytná. Pro globální pracovní jednotku, která **nezahrnovala** trvalé aktualizace databázových prostředků, neexistují žádná volání XA pro databázi **no** . Pro globální pracovní jednotku, která **má** zahrnovala tyto trvalé aktualizace, jsou volání následující:

- `xa_end` z aplikačního procesu pro deklarování konce globální transakce.
- `xa_prepare`, `xa_commit`a `xa_rollback` z procesu agenta správce front amqzlaa0. Ty se používají k dodání výsledku globální pracovní jednotky: rozhodnutí o potvrzení nebo odvolání transakce.

Má-li dynamická registrace fungovat, je životně důležité, aby databáze obsahovala způsob, jak sdělit produktu WebSphere MQ trvalou aktualizaci, kterou chce zahrnout do aktuální globální jednotky práce. Produkt WebSphere MQ poskytuje pro tento účel funkci `ax_reg` .

Kód klienta databáze, který se spouští ve vašem aplikačním procesu, vyhledá funkci `ax_reg` a volá ji, aby *dynamicky registroval* fakt, že provedla trvalou práci v rámci aktuální globální pracovní jednotky. V reakci na toto volání příkazu `ax_reg` se databáze WebSphere MQ zúčastnila, že se databáze zúčastnila. Pokud se jedná o první volání `ax_reg` v tomto připojení WebSphere MQ , zavolá proces agenta správce front `xa_open`.

Kód klienta databáze provede toto volání produktu `ax_reg` , když je spuštěn ve vašem procesu, například během volání SQL UPDATE nebo bez ohledu na volání v rozhraní API klienta databáze.

Chybové stavy

Při dynamické registraci XA existuje možnost matoucí chyby ve správci front.

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Obecným příkladem je to, že jste před spuštěním správce front zapomněli nastavit správně proměnné prostředí databáze, volání správce front na `xa_open` se nezdaří. Nelze použít žádné globální jednotky práce.

Chcete-li se tomu vyhnout, před spuštěním správce front se ujistěte, že jste nastavili příslušné proměnné prostředí. Zkontrolujte dokumentaci k databázovému produktu a rady uvedené v části “Konfigurace Db2” na stránce 49, “Konfigurace Oracle” na stránce 51a “Sybase Konfigurace” na stránce 56.

Se všemi databázovými produkty zavolá správce front `xa_open` jednou při spuštění správce front jako součást relace obnovy (jak je vysvětleno v části “Aspekty při ztrátě kontaktu se správcem prostředků XA” na stránce 59). Toto volání příkazu `xa_open` se nezdaří, pokud jste nesprávně nastavili proměnné prostředí databáze, ale nespustí se spuštění správce front. Důvodem je skutečnost, že knihovna klienta databáze používá stejný kód chyby produktu `xa_open` k označení, že je databázový server nedostupný. Produkt WebSphere MQ tuto chybu nezakládá jako závažnou chybu, protože správce front musí být schopen pokračovat ve zpracování dat mimo globální jednotky práce zahrnující danou databázi.

Subsequent calls to `xa_open` are made from the queue manager during the first MQBEGIN on a WebSphere MQ connection (if dynamic registration is not being used) or during a call by the database client code to the WebSphere MQ-provided `ax_reg` function (if dynamic registration is being used).

Časování všech chybových stavů (nebo občas FFST sestav) závisí na tom, zda používáte dynamickou registraci:

- Pokud používáte dynamickou registraci, volání MQBEGIN může být úspěšné, ale volání databáze SQL UPDATE (nebo podobné) se nezdaří.
- Pokud nepoužíváte dynamickou registraci, volání MQBEGIN selže.

Ujistěte se, že vaše proměnné prostředí jsou správně nastaveny ve vašich procesech aplikace a správce front.

Sumarizace volání XA
Zde je uveden seznam volání funkcí XA v knihovně databázových klientů v důsledku různých volání MQI, která řídí globální jednotky práce. Nejedná se o úplný popis protokolu popsaného ve specifikaci XA. Je uveden jako stručný přehled.

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Všimněte si, že volání `xa_start` a `xa_end` jsou vždy volány kódem WebSphere MQ v aplikačním procesu, zatímco `xa_prepare`, `xa_commit`a `xa_rollback` jsou vždy volány z procesu agenta správce front amqzlaa0.

Volání `xa_open` a `xa_close` zobrazené v této tabulce jsou všechny vytvořené z procesu aplikace. Proces agenta správce front volá `xa_open` za okolností popsaných v tématu “Chybové stavy” na stránce 64.

Tabulka 7. Souhrn volání funkcí XA

Volání rozhraní MQI	**Volání XA provedená s dynamickou registrací**	**Volání XA provedená bez dynamické registrace**
První MQBEG	`xa_open`	`xa_open` `xa_start`
Následné MQBEG	Žádná volání XA	`xa_start`

Tabulka 7. Souhrn volání funkcí XA (pokračování)

Volání rozhraní MQI	**Volání XA provedená s dynamickou registrací**	**Volání XA provedená bez dynamické registrace**

MQCMIT (**bez** `ax_reg` je volána během aktuální globální pracovní jednotky)	Žádná volání XA	`xa_end`
		`xa_prepare`
		`xa_commit`
		`xa_rollback`

MQCMIT (**s** `ax_reg` probíhá volání v rámci aktuální globální pracovní jednotky)	`xa_end`
	`xa_prepare`
	`xa_commit`
	`xa_rollback`
		Nepoužívá se. V nedynamickém režimu nejsou žádná volání funkce ax_reg.

MQBACK (**bez** `ax_reg` je volán během aktuální globální pracovní jednotky)	Žádná volání XA	`xa_end` `xa_rollback`
MQBACK (**s** `ax_reg` probíhá volání v rámci aktuální globální pracovní jednotky)	`xa_end` `xa_rollback`	Nepoužívá se. V nedynamickém režimu nejsou žádná volání funkce ax_reg.
MQDISC, kde bylo volání MQCMIT nebo MQBACK voláno jako první. Pokud tomu tak nebylo, zpracování MQCMIT se nejprve provede během operace MQDISC.	`xa_close`	`xa_close`

Notes:

1. Pro MQCMIT je `xa_commit` volán, pokud je `xa_prepare` úspěšný. Jinak se volá `xa_rollback` .

Scénář 2: Další software poskytuje koordinaci

Ve scénáři 2 externí správce transakcí koordinuje globální jednotky práce, spouští a potvrzuje je pod kontrolou rozhraní API správce transakcí. Přísloví MQBEGIN, MQCMIT a MQBACK jsou nedostupné.

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Tento oddíl popisuje tento scénář, včetně:

- “Koordinace externího bodu synchronizace” na stránce 66
- “Použití systému CICS” na stránce 69
- “Použití serveru Microsoft Transaction Server (COM +)” na stránce 74

Klient IBM WebSphere MQ pro HP Integrity NonStop Server může použít nástroj HP NonStop Transaction Management Facility (TMF) ke koordinaci globálních transakcí. Další informace najdete v tématu Použití systému HP NonStop TMF.

Koordinace externího bodu synchronizace

Globální pracovní jednotka může být také koordinována externím správcem transakcí standardu X/Open XA. Zde se správce front produktu WebSphere MQ podílí na práci, ale nekoordinuje danou pracovní jednotku.

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Tok řízení v rámci globální transakce koordinovaný externím správcem transakcí je následující:

1. Aplikace říká koordinátorovi externího bodu synchronizace (například TXSeries), že chce spustit transakci.
2. Koordinátor synchronizačních bodů informuje známé správce prostředků, například WebSphere MQo aktuální transakci.
3. Aplikace zadá volání správci prostředků přidruženému k aktuální transakci. Aplikace může například vydávat volání `MQGET` na WebSphere MQ.
4. Aplikace vydá do koordinátora externího synchronizačního bodu požadavek na potvrzení nebo odvolání.
5. Koordinátor synchronizačních bodů dokončí transakci zadáním příslušných volání do jednotlivých správců prostředků, obvykle pomocí protokolů s dvoufázovým potvrzováním.

Podporované úrovně externích koordinátorů bodu synchronizace, které mohou poskytnout proces dvoufázového potvrzení pro transakce, v nichž je produkt WebSphere MQ zapojen, jsou definovány na adrese Podrobné systémové požadavky produktuIBM WebSphere MQ.

Zbytek této sekce popisuje, jak povolit externí jednotky práce.

Struktura přepínače IBM WebSphere MQ XA
Každý správce prostředků, který se podílí na externě koordinované jednotce práce, musí poskytovat strukturu přepínačů XA. Tato struktura definuje jak schopnosti správce prostředků, tak funkce, které mají být volány koordinátorem synchronizačního bodu.

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Produkt IBM WebSphere MQ poskytuje dvě verze této struktury:

- *`MQRMIXASwitch`* pro statickou správu prostředků XA
- *`MQRMIXASwitchDynamic`* pro dynamickou správu prostředků XA

Informace o tom, zda použít statické nebo dynamické rozhraní správy prostředků, zjistíte v dokumentaci ke správci transakcí. Kdykoli ji správce transakcí podporuje, doporučujeme vám používat dynamickou správu prostředků XA.

Někteří 64bitoví správci transakcí zpracují typ *long* ve specifikaci XA jako 64bitovou a někteří s ní pracují jako 32bitový. Produkt WebSphere MQ podporuje oba modely:

- Je-li správce transakcí 32bitový nebo je-li správce transakcí 64bitový, ale zachází s typem *long* jako 32bitovou, použijte soubor LOAD uvedený v části Tabulka 8 na stránce 67.
- Pokud je správce transakcí 64bitový a pracuje s typem *long* jako 64bitový, použijte soubor LOAD s tímto souborem, který je uveden v části Tabulka 9 na stránce 68.

Seznam známých 64bitových správců transakcí, který s typem *long* zachází jako s 64bitovým systémem, je uveden v tématu Tabulka 10 na stránce 68. Pokud si nejste jisti, který model používá správce transakcí, prostudujte si dokumentaci ke správci transakcí.

Tabulka 8. Názvy souborů načtení přepínače XA

Platforma	**Název zaváděcího souboru přepínače (server)**	**Název zaváděcího souboru přepínače (rozšířený transakční klient)**
Windows	*mqmxa.dll*	*mqcxa.dll*
AIX (bez podprocesů)	*libmqmxa.a*	*libmqcxa.a*

Tabulka 8. Názvy souborů načtení přepínače XA (pokračování)		
Platforma	**Název zaváděcího souboru přepínače (server)**	**Název zaváděcího souboru přepínače (rozšířený transakční klient)**
AIX (s podporou podprocesů)	*libmqmxa_r.a*	*libmqcxa_r.a*
HP-UX (bez podprocesů)	*libmqmxa.so*	*libmqcxa.so*
HP-UX (s podporou podprocesů)	*libmqmxa_r.so*	*libmqcxa_r.so*
Linux (bez podprocesů)	*libmqmxa.so*	*libmqcxa.so*
Linux (s podporou podprocesů)	*libmqmxa_r.so*	*libmqcxa_r.so*
Solaris	*libmqmxa.so*	*libmqcxa.so*

Tabulka 9. Alternativní 64bitové názvy zaváděcích souborů přepínače XA		
Platforma	**Název zaváděcího souboru přepínače (server)**	**Název zaváděcího souboru přepínače (rozšířený transakční klient)**
AIX (bez podprocesů)	*libmqmxa64.a*	*libmqcxa64.a*
AIX (s podporou podprocesů)	*libmqmxa64_r.a*	*libmqcxa64_r.a*
HP-UX (bez podprocesů)	*libmqmxa64.so*	*libmqcxa64.so*
HP-UX (s podporou podprocesů)	*libmqmxa64_r.so*	*libmqcxa64_r.so*
Linux (bez podprocesů)	*libmqmxa64.so*	*libmqcxa64.so*
Linux (s podporou podprocesů)	*libmqmxa64_r.so*	*libmqcxa64_r.so*
Solaris	*libmqmxa64.so*	*libmqcxa64.so*

Tabulka 10. 64bitový správce transakcí, který vyžaduje alternativní 64bitový zaváděcí soubor přepínače
správce transakcí
Tuxedo

Někteří externí koordinátoři bodu synchronizace (nikoli CICS) vyžadují, aby se každý správce prostředků, který se podílí na transakci, dodává jeho název v poli názvu struktury přepínače XA. Název správce prostředků produktu WebSphere MQ je MQSeries_XA_RMI.

Koordinátor synchronizačních bodů definuje, jak se k němu odkazuje struktura přepínače WebSphere MQ XA. Informace o propojování struktury přepínačů WebSphere MQ XA se systémem CICS jsou k dispozici v produktu “Použití systému CICS” na stránce 69. Informace o propojování struktury přepínače WebSphere MQ XA s dalšími koordinátory bodu synchronizace standardu XA naleznete v dokumentaci dodávané s těmito produkty.

Při použití produktu WebSphere MQ se všemi koordinátory bodu synchronizace standardu XA se vztahují následující pokyny:

- Struktura xa_info předávaná při každém volání xa_open od koordinátora synchronizačních bodů zahrnuje název správce front WebSphere MQ . Název má stejný tvar jako název správce front předaný do volání `MQCONN` . Je-li název předaný v volání xa_open prázdný, použije se výchozí správce front.

  Alternativně může struktura xa_info obsahovat hodnoty pro parametry *TPM* a *AXLIB* . Parametr *TPM* uvádí, který správce transakcí se používá. Platné hodnoty jsou CICS, TUXEDO a ENCINA. Parametr *AXLIB* uvádí název knihovny, která obsahuje funkce ax_reg a ax_unreg správce transakcí. Další informace o těchto parametrech naleznete v tématu Konfigurace rozšířeného transakčního klienta. Pokud struktura xa_info obsahuje některý z těchto parametrů, je název správce front zadán v parametru *QMNAME* , pokud není použit výchozí správce front.
- Pouze jeden správce front se může v daném okamžiku podílet na transakci koordinované instancí externího koordinátora synchronizačního bodu. Koordinátor synchronizačních bodů je efektivně připojen ke správci front a je podřízen pravidlu, že je podporováno pouze jedno připojení v daném okamžiku.
- Všechny aplikace, které zahrnují volání do externího koordinátora synchronizačních bodů, se mohou připojit pouze ke správci front, který se podílí na transakci spravované externím koordinátorem (protože jsou již k tomuto správci front efektivně připojeni). Tyto aplikace však musí při volání obslužné rutiny připojení a volání `MQDISC` před jejich ukončením vydat volání `MQCONN` .
- Správce front s aktualizacemi prostředků koordinovaným externím koordinátorem bodu synchronizace musí být spuštěn před koordinátorem externího bodu synchronizace. Podobně musí koordinátor synchronizačního bodu skončit před správcem front.
- Pokud se koordinátor externího synchronizačního bodu ukončí nestandardně, zastavte a znovu spusťte správce front ***před*** opětovným spuštěním koordinátora synchronizačních bodů, abyste se ujistili, že všechny operace systému zpráv nepotvrzené v době selhání budou řádně vyřešeny.

Použití systému CICS

CICS je jedním z prvků TXSeries.

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Verze produktu TXSeries , které odpovídají standardu XA (a používají proces s dvoufázovým potvrzováním), jsou definovány na adrese: Podrobné systémové požadavky produktuIBM WebSphere MQ

Produkt WebSphere MQ také podporuje ostatní správce transakcí. Aktuální seznamy podporovaných softwaru viz Podrobné systémové požadavky produktuIBM WebSphere MQ .

Požadavky na proces dvoufázového potvrzování

Požadavky na proces dvoufázového potvrzování při použití procesu dvoufázového potvrzení CICS s produktem WebSphere MQ. Tyto požadavky se nevztahují na z/OS.

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Všimněte si následujících požadavků:

- WebSphere MQ a CICS musí být umístěny na stejném fyzickém počítači.
- Produkt WebSphere MQ nepodporuje klienta CICS na klientu WebSphere MQ MQI.
- Musíte spustit správce front se svým názvem uvedeným v sekci definice prostředku XAD, ***před*** pokusem o spuštění CICS. Pokud jste nepřidali sekci definice prostředku XAD pro produkt WebSphere MQ do oblasti CICS , dojde k selhání při spuštění systému CICS .
- Pouze jeden správce front produktu WebSphere MQ je dostupný v době z jediné oblasti CICS .
- Transakce CICS musí před přístupem k prostředkům produktu WebSphere MQ vydat požadavek `MQCONN` . Volání `MQCONN` musí určovat název správce front WebSphere MQ určeného v objektu stanza

XAOpen v sekci definice prostředku XAD pro oblast CICS . Je-li tato položka prázdná, musí požadavek MQCONN určovat výchozího správce front.

- Transakce CICS , která přistupuje k prostředkům produktu WebSphere MQ , musí před návratem do CICSzadat volání MQDISC z transakce. Pokud to neuděláte, může to znamenat, že aplikační server CICS je stále připojen a ponechá fronty otevřené. Navíc, pokud neinstalujete uživatelskou proceduru ukončení úlohy (viz “Ukázková uživatelská procedura ukončení úlohy” na stránce 73), může být aplikační server CICS později ukončen nestandardně, pravděpodobně během následné transakce.
- Musíte zajistit, aby ID uživatele CICS (cnics) bylo členem skupiny mqm, aby měl kód CICS oprávnění volat produkt WebSphere MQ.

  U transakcí spuštěných v prostředí CICS správce front přizpůsobuje své metody autorizace a určuje kontext následujícím způsobem:

  – Správce front se dotáže na ID uživatele, pod kterým produkt CICS spustí transakci. Jedná se o ID uživatele, které je kontrolováno správcem oprávnění objektu, a používá se pro kontextové informace.
  – V kontextu zprávy je typ aplikace MQAT_CICS.
  – Název aplikace v kontextu je zkopírován z názvu transakce CICS .

Obecná podpora standardu XA
Obecná podpora XA není v systému IBM ipodporována. Je k dispozici modul pro načtení přepínače XA, který umožňuje propojit produkt CICS s produktem WebSphere MQ v systémech UNIX and Linux . Kromě toho jsou k dispozici ukázkové soubory zdrojového kódu, které vám umožní vývoj přepínačů XA pro další zprávy transakcí.

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Názvy dodávaných modulů načtení přepínače jsou:

Tabulka 11. Základní kód pro aplikace CICS : inicializační rutina XA

C (zdroj)	**C (exec)-přidání jedné z následujících možností do XAD.Stanza**
amqzscix.c	amqzsc- TXSeries pro AIX, verze 5.1, amqzsc- TXSeries pro HP-UX, verze 5.1 amqzsc- TXSeries for Sun Solaris, verze 5.1
amqzscin.c	mqmc4swi - TXSeries pro Windows, verze 5.1

Sestavování knihoven pro použití s produktem TXSeries for Multiplatforms
Tyto informace použijte při sestavování knihoven pro použití s produktem TXSeries for Multiplatforms.

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Předpřipravené soubory načtení přepínače jsou sdílené knihovny (nazývané *DLL* v systému Windows), které můžete použít s programy CICS, které vyžadují dvoufázovou transakci s potvrzením pomocí protokolu XA. Názvy těchto předpřipravených knihoven jsou v tabulce Základní kód pro aplikace CICS : inicializační rutina XA. Ukázkový zdrojový kód je také dodáván v následujících adresářích:

Tabulka 12. Instalační adresáře v operačních systémech Windows, UNIX and Linux		
Platforma	**Adresář**	**Zdrojový soubor**
UNIX and Linux	*MQ_INSTALLATION_PATH*/ samp/	amqzscix.c
Windows	*MQ_INSTALLATION_PATH*\Tools \c \ Samples	amqzscin.c

kde*MQ_INSTALLATION_PATH* je adresář, do kterého jste nainstalovali produkt IBM WebSphere MQ.

Chcete-li sestavit zaváděcí soubor přepínače z ukázkového zdroje, postupujte podle pokynů pro váš operační systém:

AIX

Spusťte následující příkaz:

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

Solaris

Spusťte následující příkaz:

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

HP-UX

Spusťte následující příkaz:

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

Platformy Linux

Spusťte následující příkaz:

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

Windows

Postupujte takto:

1. Použijte příkaz cl k sestavení amqzscin.obj tak, že kompilujete alespoň následující proměnné:

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

2. Vytvořte definiční soubor modulu s názvem `mqmc1415.def`, který obsahuje následující řádky:

   ```
   LIBRARY MQMC4SWI
   EXPORTS
   CICS_XA_Init
   ```

3. Pomocí příkazu **lib** sestavíte soubor exportu a knihovnu importu s použitím alespoň této volby:

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

   Je-li příkaz lib úspěšný, sestaví se také soubor `mqmc4swi.exp` .

4. Použijte příkaz link k sestavení `mqmc4swi.dll` s pomocí alespoň této volby:

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

IBM WebSphere MQ Podpora XA a Tuxedo
IBM WebSphere MQ on Okna,UNIX and Linux systems can block Tuxedo-coordinated XA applications indefinitely in xa_start.

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

K tomu může dojít pouze v případě, že se dva nebo více procesů koordinovaných produktem Tuxedo v rámci jedné globální transakce pokusí o přístup k produktu IBM WebSphere MQ s použitím stejného ID větve transakce (XID). Pokud produkt Tuxedo poskytuje každému procesu v globální transakci jiný identifikátor XID, který má být použit s produktem IBM WebSphere MQ, k této operaci nelze provést.

Chcete-li se vyhnout problému, nakonfigurujte každou aplikaci v produktu Tuxedo, která přistupuje k produktu IBM WebSphere MQ pod jedním globálním ID transakce (gtrid), do své vlastní skupiny serverů Tuxedo. Procesy ve stejné skupině serverů používají stejný identifikátor XID při přístupu ke správcům prostředků v zastoupení jediné gtrid a jsou proto náchylné k blokování v xa_start v produktu IBM WebSphere MQ. Procesy v různých skupinách serverů používají oddělené identifikátory XID při přístupu ke správcům prostředků a nemusí tedy serializovat svou práci transakce v produktu IBM WebSphere MQ.

Povolení procesu dvoufázového potvrzování CICS
Chcete-li povolit CICS používat dvoufázový proces potvrzování pro koordinaci transakcí, které zahrnují volání MQI, přidejte do oblasti CICS záznam objektu stanza definice prostředku CICS XAD. Všimněte si, že toto téma není použitelné pro z/OS.

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Zde je uveden příklad přidání položky objektu stanza XAD pro produkt WebSphere MQ for Windows, kde `<Drive>` je jednotka, kde je nainstalován produkt WebSphere MQ (například D:).

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

V případě rozšířených transakčních klientů použijte soubor LOAD mqcc4swi.dll.

Zde je uveden příklad přidání položky objektu stanza XAD pro systémy WebSphere MQ for UNIX and Linux , kde *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ instalován:

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

Pro rozšířené transakční klienty použijte soubor LOAD amqczsc.

Informace o použití příkazu **cicsadd** naleznete v příručce *CICS Administration Reference*nebo v příručce *CICS Administration Guide* pro vaši platformu.

Volání do produktu WebSphere MQ lze zahrnout do transakce CICS a prostředky produktu WebSphere MQ budou potvrzeny nebo odvolány podle pokynů v systému CICS. Tato podpora není k dispozici pro klientské aplikace.

musíte vydat příkaz MQCONN z transakce CICS za účelem přístupu k prostředkům produktu WebSphere MQ následovaným odpovídajícím parametrem MQDISC při ukončení.

Povolení uživatelských procedur CICS
Uživatelská procedura CICS *point* (obvykle označovaná jako *uživatelská procedura*) je místem v modulu CICS , na kterém může CICS přenášet řízení do programu, který jste napsali (uživatelský ukončovací program *program*), a na který CICS může pokračovat v řízení, až váš uživatelský program dokončí svou práci.

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Než použijete uživatelskou proceduru CICS , přečtěte si příručku *CICS Administration Guide* pro vaši platformu.

Ukázková uživatelská procedura ukončení úlohy
WebSphere MQ dodá vzorový zdrojový kód pro ukončení úlohy CICS pro ukončení úlohy.

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Ukázkový zdrojový kód se nachází v následujících adresářích:

Tabulka 13. Ukončení ukončení úlohy CICS

Platforma	**Adresář**	**Zdrojový soubor**
Systémy UNIX and Linux	*MQ_INSTALLATION_PATH*/samp	amqzscgx.c
Windows	*MQ_INSTALLATION_PATH*\Tools \c \ Samples	amqzscgn.c

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

Pokyny pro sestavení ukázkové uživatelské procedury ukončení úlohy jsou obsaženy v komentářích v horní části každého zdrojového souboru.

Tato uživatelská procedura je vyvolána systémem CICS při normálním a abnormálním ukončení úlohy (po provedení libovolného bodu synchronizace). V ukončovacím programu není povolena žádná obnovitelná práce.

Tyto funkce jsou používány pouze v kontextu produktu WebSphere MQ a CICS , ve kterém verze CICS podporuje rozhraní XA. CICS odkazuje na tyto knihovny jako "programy" nebo "uživatelské procedury".

CICS má definovaný počet uživatelských procedur a amqzscgx, pokud je použit, je definován a povolen v systému CICS jako "Uživatelská procedura ukončení úlohy (UE014015)", tj. výstupní číslo 15.

Je-li uživatelská procedura ukončení úlohy volána systémem CICS, produkt CICS již informoval produkt WebSphere MQ o stavu ukončení úlohy a produkt WebSphere MQ provedl příslušnou akci (potvrdit nebo odvolat). Veškerý výstup má vydat příkaz MQDISC k vyčištění.

Jeden účel instalace a konfigurace systému CICS pro použití ukončení ukončení úlohy slouží k ochraně systému před některými důsledky chybného kódu aplikace. Je-li například transakce CICS ukončena nestandardně bez volání funkce MQDISC a není nainstalována žádná uživatelská procedura ukončení úlohy, může dojít k následnému nezotavitelnému selhání oblasti CICS (během přibližně 10 sekund). Důvodem je skutečnost, že podproces WebSphere MQ, který se spouští v procesu cicsas, nebude odeslán

a poskytnut čas k vyčištění a návratu. Symptomy mohou být, že proces cicsas končí okamžitě, přičemž napsaly sestavy FFST do adresáře /var/mqm/errors nebo ekvivalentní umístění v systému Windows.

Použití serveru Microsoft Transaction Server (COM +)

COM + (Microsoft Transaction Server) je navržen tak, aby pomáhal uživatelům spouštět aplikace obchodní logiky v typickém serveru střední vrstvy.

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Důležité informace naleznete v příručce Funkce, které lze použít pouze s primární instalací na systému Windows .

COM + rozděluje práci na *aktivity*, které jsou obvykle krátké nezávislé diskové bloky obchodní logiky, jako např. *převod finančních prostředků z účtu A na účet B*. COM + silně spoléhá na orientaci objektů, a zejména na COM; aktivita COM + je volně zastoupena objektem COM (obchodní objekt).

COM + je integrovaná část operačního systému. Chcete-li použít COM + na systémech Windows 2000 a Windows XP, potřebujete Hotfix Q313582 (také známý jako kumulativní balík COM + 19.1).

Produkt COM + poskytuje tři služby administrátorovi obchodních objektů a odstraňuje velkou část obav od programátora obchodních objektů:

- Správa transakcí
- Zabezpečení
- Fondy prostředků

Obvykle používáte model COM + s předřazeným kódem, který je klientem COM k objektům v rámci COM + a back-endových služeb, jako je databáze, s přemostění WebSphere MQ mezi obchodním objektem COM + a back-endem.

Front-endový kód může být samostatný program nebo ASP (Active Server Page) hostované serverem Microsoft Internet Information Server (IIS). Front-endový kód může být na stejném počítači jako COM + a jeho obchodní objekty, s připojením přes COM. Alternativně může být front-endový kód na jiném počítači, s připojením přes DCOM. V různých situacích můžete použít různé klienty pro přístup ke stejnému obchodnímu objektu COM +.

Back-endový kód může být na stejném počítači jako COM + a jeho obchodní objekty, nebo na jiném počítači s připojením prostřednictvím některého z podporovaných protokolů WebSphere MQ .

Ukončení platnosti globálních jednotek práce

Správce front může být konfigurován tak, aby vypršelo globální jednotky práce po předkonfigurovaném intervalu nečinnosti.

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Chcete-li toto chování povolit, nastavte následující proměnné prostředí:

- *AMQ_TRANSACTION_EXPIRY_RESCAN*= < interval opětovného zpracování (v milisekundách) >
- *AMQ_XA_TRANSACTION_EXPIRY*= < interval vypršení časového limitu v milisekundách >

Upozornění: Proměnné prostředí ovlivňují pouze transakce, které jsou ve stavu *Nečinný* v tabulce 6-4 specifikace XA. To znamená, že transakce, které nejsou přidruženy k žádnému podprocesu aplikace, ale pro které externí software Transaction Manager ještě nevolali volání funkce **`xa_prepare`** .

Externí správci transakcí udržují pouze protokol transakcí, které jsou připraveny, potvrzeny nebo vráceny do původního stavu. Pokud správce externích transakcí z jakéhokoli důvodu odejde z jakéhokoliv důvodu, vrátí připravené, potvrzené a odvolané transakce k dokončení, ale všechny aktivní transakce, které se dosud připravovaly, se stanou osiřelými. Chcete-li se této chybě vyhnout, nastavte parametr

AMQ_XA_TRANSACTION_EXPIRY tak, aby umožňoval očekávaný interval mezi aplikací, která provádí rozhraní MQI transakčních rozhraní API, a dokončení transakce, jež provedla transakční práci na jiných správcích prostředků.

Chcete-li zajistit včasné vyčištění po vypršení platnosti *AMQ_XA_TRANSACTION_EXPIRY* , nastavte hodnotu *AMQ_TRANSACTION_EXPIRY_RESCAN* na nižší hodnotu, než je interval *AMQ_XA_TRANSACTION_EXPIRY* , v ideálním případě tak, aby se opětovné skenování vyskytčilo v intervalu *AMQ_XA_TRANSACTION_EXPIRY* více než jednou.

Dispozice jednotky zotavení

Produkt WebSphere MQ for z/OS poskytuje jednotku dispozic zotavení. Tato funkce vám umožňuje konfigurovat, zda lze při připojení k jinému správci front v rámci stejné skupiny sdílení front (QSG) řídit druhou fázi transakcí s dvoufázovým potvrzováním (například během zotavení).

Poznámka: Toto téma je také k dispozici v produktu IBM MQ Version 8.0 a v novějších verzích. Avšak nemůžete přepnout na pozdější verzi pomocí okénka se seznamem "Změnit verzi" . Chcete-li přejít na téma v pozdější verzi, upravte číslo verze v poli Adresa URL ve vašem prohlížeči.

Produkt WebSphere MQ for z/OS V7.0.1 a novější podporuje dispozice jednotky zotavení.

Dispozice jednotky zotavení

Dispozice jednotky zotavení souvisí s připojením aplikace a následně ke všem transakcím, které spouští. Existují dvě možné dispozic jednotky obnovy.

- Skupina GROUP obnovy zotavení identifikuje, že transakční aplikace je logicky připojena ke skupině sdílení front a nemá afinitu k žádnému konkrétnímu správci front. Všechny transakce s dvoufázovým potvrzováním, které začínají s dokončením phase-1 procesu odevzdání, tj. jsou nejisté, mohou být dotazovány a vyřešeny, jsou-li připojeny k libovolnému správci front v rámci skupiny QSG. Ve scénáři zotavení to znamená, že koordinátor transakcí se nemusí znovu připojit ke stejnému správci front, což může být nedostupné.
- Dispozice QMGR odebrání zotavení identifikuje, že aplikace má přímou afinitu ke správci front, ke kterému je připojen, a všechny transakce, které tento proces spouští, mají také tuto dispozice.

  V případě scénáře zotavení se koordinátor transakcí musí znovu připojit ke stejnému správci front, aby se dotazoval a vyřešil všechny neověřené transakce bez ohledu na to, zda správce front náleží do skupiny sdílení front.

Rozhodování o tom, jaký programovací jazyk použít

Tyto informace použijte k vyhledání informací o programovacích jazycích a rámcích podporovaných produktem IBM WebSphere MQa o některých aspektech jejich použití.

Produkt IBM WebSphere MQ poskytuje podporu pro následující procedurální jazyky programování:

- C
- Visual Basic (pouze systémy Windows)
- COBOL

Tyto jazyky používají rozhraní fronty zpráv (MQI) pro přístup ke službám front zpráv. Další informace o podpoře těchto jazyků najdete v tématu “Použití procedurálních jazyků s produktem WebSphere MQ” na stránce 76.

Produkt IBM WebSphere MQ poskytuje podporu pro:

- .NET
- ActiveX
- JAZYK C++
- Java
- JMS

Tyto jazyky používají objektový model produktu IBM WebSphere MQ , který poskytuje třídy, které poskytují stejné funkce jako volání a struktury produktu WebSphere MQ , ale které jsou přirozenější způsobem programování v objektově orientovaném prostředí. Některé z jazyků, které používají objekt IBM WebSphere MQ Object Model, poskytují další funkce, které nejsou k dispozici v rozhraní MQI (Message Queue Interface). Další informace o podpoře těchto jazyků najdete v tématu “Programování v objektově orientovaném prostředí s produktem WebSphere MQ” na stránce 76.

Použití procedurálních jazyků s produktem WebSphere MQ

Podrobné informace o tom, jak psát vaše aplikace ve zvoleném jazyce, najdete v následujících odkazech:

- “Cování v C” na stránce 79
- “Kódování ve Visual Basicu” na stránce 83
- “Kódování v jazyce COBOL” na stránce 82

Přehled rozhraní volání pro procedurální jazyky naleznete v tématu Popisy volání. Toto téma obsahuje seznam volání MQI a každé volání ukazuje, jak kódovat volání v každém z těchto jazyků.

Produkt WebSphere MQ poskytuje soubory definic dat, které vám pomohou s napsáním aplikací. Úplný popis viz “Soubory definic dat produktu IBM WebSphere MQ” na stránce 77.

Pokud si můžete zvolit jazyk, ve kterém se mají programy kódovat, vezměte v úvahu maximální délku zpráv, které budou programy zpracovávat. Pokud budou vaše programy zpracovávat pouze zprávy o známé maximální délce, můžete je kódovat v kterémkoli z podporovaných programovacích jazyků. Pokud však neznáte maximální délku zpráv, které budou programy muset zpracovat, bude jazyk, který zvolíte, záviset na tom, zda píšete CICS, IMSnebo dávkovou aplikaci:

IMS a dávka
Kódujte programy v jazycích C, PL/I nebo assembler tak, aby používaly funkce těchto jazyků k získání a uvolnění libovolných množství paměti. Alternativně můžete kódovat své programy v jazyce COBOL, ale používat podprogramy jazyka v jazyku assembler, PL/I nebo C k získání a uvolnění paměti.

CICS
Kódujte programy v libovolném jazyce, který je podporován systémem CICS. Rozhraní EXEC CICS poskytuje volání pro správu paměti, je-li to nutné.

Programování v objektově orientovaném prostředí s produktem WebSphere MQ

Některé z jazyků a programovacích rámců, které používají model objektů produktu IBM WebSphere MQ , poskytují další funkce, které nejsou k dispozici v rozhraní MQI (Message Queue Interface). Podrobnosti o třídách, metodách a vlastnostech poskytovaných objektem IBM WebSphere MQ Object Model najdete v tématu “Objektový model produktu IBM WebSphere MQ” na stránce 84.

.NET
Informace o kódování programů .NET pomocí tříd .NET produktu WebSphere MQ naleznete v příručce Použití rámce .NET . Message Service Clients for C/C++ and .NET poskytují rozhraní API s názvem XMS , které má stejnou sadu rozhraní jako služba JMS (Java Message Service). API.

JAZYK C++
Produkt IBM WebSphere MQ poskytuje třídy C + + ekvivalentní objektům WebSphere MQ a některé další třídy ekvivalentní k datovým typům pole. Poskytuje řadu funkcí, které nejsou prostřednictvím rozhraní MQI k dispozici. Informace o kódovacích programech pomocí objektového modelu produktu WebSphere MQ v jazycích C + + viz Použití C++ . Message Service Clients for C/C++ and .NET poskytuje rozhraní API (Application Programming Interface) s názvem XMS , které má stejnou sadu rozhraní jako služba JMS (Java Message Service). API.

Java
Informace o kódování programů pomocí objektového modelu WebSphere MQ v prostředí Java naleznete v příručce Použití jazyka Java . Informace o rozdílech mezi třídami IBM WebSphere MQ pro třídy Java a IBM WebSphere MQ , které vám pomohou rozhodnout se, které použít, viz “Měla bych používat třídy IBM WebSphere MQ pro třídy Java nebo IBM WebSphere MQ pro platformu JMS?” na stránce 86.

JMS

Produkt Websphere MQ také poskytuje třídy, které implementují specifikaci JMS (Java Message Service). Podrobné informace o třídách Websphere MQ pro platformu JMS naleznete v tématu Použití jazyka Java. Informace o rozdílech mezi třídami IBM WebSphere MQ pro jazyk Java a třídami IBM WebSphere MQ , které vám pomohou rozhodnout se, které použít, viz “Měla bych používat třídy IBM WebSphere MQ pro třídy Java nebo IBM WebSphere MQ pro platformu JMS?” na stránce 86.

Message Service Clients for C/C++ and .NET poskytují rozhraní API s názvem XMS , které má stejnou sadu rozhraní jako služba JMS (Java Message Service). API.

ActiveX

Produkt WebSphere MQ ActiveX je obecně znám jako MQAX. MQAX je součástí produktu WebSphere MQ for Windows. Podpora pro ActiveX byla stabilizována na úrovni WebSphere MQ verze 6.0 . Chcete-li využívat funkce zavedené do produktu WebSphere MQ novější než verze 6.0, zvažte místo toho použití prostředí .NET. Informace o kódování programů pomocí objektu WebSphere MQ Object Model v ActiveXnaleznete v části Použití rozhraní modelu COM (Component Object Model) (WebSphere MQ Automation Classes for ActiveX) .

Související pojmy

Technický přehled

“Vývoj aplikací” na stránce 7

Produkt IBM WebSphere MQ nabízí několik způsobů, jak vyvíjet aplikace k odesílání a přijímání zpráv, které potřebujete k podpoře vašich obchodních procesů. Také můžete vyvíjet aplikace pro správu správců front a souvisejících prostředků.

“Koncepty vývoje aplikací” na stránce 7

K zápisu aplikací IBM WebSphere MQ můžete použít volbu procedurálních nebo objektově orientovaných jazyků. Informace o konceptech produktu IBM WebSphere MQ , které jsou užitečné pro vývojáře aplikací, naleznete v odkazech v tomto tématu.

Související odkazy

Odkaz na vývoj aplikací

Soubory definic dat produktu IBM WebSphere MQ

Produkt IBM WebSphere MQ poskytuje soubory definic dat, které vám pomohou psát vaše aplikace.

Soubory definice dat jsou také známé jako:

Jazyk	Definice dat
C	Zahrnout soubory nebo hlavičkové soubory
Visual Basic	Soubory modulu (pouze 32bitové verze)
COBOL	Kopírovat soubory
Asembler	Makra
PL/I	Zahrnout soubory

Soubory definic dat, které vám pomohou při psaní uživatelských procedur, jsou popsány v části WebSphere MQ COPY, header, include a module files.

Soubory definic dat, které vám pomohou s procedurami pro psaní instalovatelných služeb, jsou popsány v tématu “Uživatelské procedury, uživatelské procedury rozhraní API a instalovatelné služby produktu WebSphere MQ” na stránce 358.

Informace o souborech definic dat podporovaných v jazyce C + + naleznete v části Použití C++.

Názvy souborů definic dat mají předponu CMQ a příponu, která je určena programovacím jazykem:

Přípona	Jazyk
a	Jazyk asembleru

Přípona	Jazyk
b	Visual Basic
c	C
l	COBOL (bez inicializovaných hodnot)
p	PL/I
v	COBOL (s výchozí sadou hodnot)

Instalační knihovna

Název **thlqual** je kvalifikátor vyšší úrovně instalační knihovny v systému z/OS.

Toto téma uvádí soubory definic dat produktu WebSphere MQ v těchto záhlavích:

- “Zahrnují soubory jazyka C” na stránce 78
- “Soubory modulu Visual Basic” na stránce 78
- “Soubory kopie COBOL” na stránce 78

Zahrnují soubory jazyka C

Soubory začlenění produktu WebSphere MQ C jsou uvedeny v seznamu Soubory záhlaví C. Jsou instalovány v následujících adresářích nebo knihovnách:

Platforma	Instalační adresář nebo knihovna
Platformy UNIX	*MQ_INSTALLATION_PATH*/inc/
systémy Windows	*MQ_INSTALLATION_PATH*\Tools\c\include

kde *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

Poznámka: Na platformách UNIX jsou soubory začlenění symbolicky propojeny do `/usr/include`.

Další informace o struktuře adresářů najdete v tématu Plánování podpory systému souborů .

Soubory modulu Visual Basic

Produkt WebSphere MQ for Windows poskytuje čtyři soubory modulu Visual Basic.

Jsou uvedeny v části Soubory modulu Visual Basic a instalovány v

```
MQ_INSTALLATION_PATH\Tools\Samples\VB\Include
```

Soubory kopie COBOL

V případě COBOL poskytuje produkt WebSphere MQ samostatné soubory kopií obsahující pojmenované konstanty a dva soubory kopií pro každou ze struktur.

Pro každou strukturu existují dva kopírovací soubory, protože každý je poskytován jak s počátečními hodnotami, tak bez počátečních hodnot:

- Ve funkci WORKING-STORAGE SECTION programu COBOL použijte soubory, které inicializují pole struktury na výchozí hodnoty. Tyto struktury jsou definovány v kopírovaných souborech, které mají příponu s příponou V (hodnoty).
- V oddíle LINKAGE v programu COBOL použijte struktury bez počátečních hodnot. Tyto struktury jsou definovány ve kopírovacích souborech, které mají příponu s písmenem L (páková).

Soubory kopie produktu WebSphere MQ COBOL jsou uvedeny v seznamu Soubory COBOL COPY. Jsou instalovány v následujících adresářích:

Platforma	Instalační adresář nebo knihovna
Ostatní platformy UNIX	*MQ_INSTALLATION_PATH*/inc/
Windows	*MQ_INSTALLATION_PATH*\Tools\cobol\copybook (for Micro Focus COBOL) *MQ_INSTALLATION_PATH*\Tools\cobol\copybook\VAcobol (pro IBM VisualAge COBOL)

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

Do svého programu zahrňte pouze ty soubory, které potřebujete. Postupujte takto s jedním nebo více příkazy COPY po deklaraci úrovně level-01 . To znamená, že v případě potřeby můžete zahrnout více verzí struktur do programu. Všimněte si, že CMQV je velký soubor.

Zde je příklad kódu COBOL pro zahrnutí souboru kopie CMQMDV:

```
01 MQM-MESSAGE-DESCRIPTOR.
   COPY CMQMDV.
```

Každá deklarace struktury začíná položkou level-01 ; můžete deklarovat několik instancí struktury zakódováním deklarace level-01 , za kterou následuje příkaz COPY, který se kopíruje ve zbytku deklarace struktury. Chcete-li se odkázat na odpovídající instanci, použijte klíčové slovo IN.

Zde je příklad kódu COBOL pro zahrnutí dvou instancí CMQMDV:

```
* Declare two instances of MQMD
 01  MY-CMQMD.
     COPY CMQMDV.
 01  MY-OTHER-CMQMD.
     COPY CMQMDV.
*
* Set MSGTYPE field in MY-OTHER-CMQMD
     MOVE MQMT-REQUEST TO MQMD-MSGTYPE IN MY-OTHER-CMQMD.
```

Srovnejte struktury na 4bajtových hranicím. Použijete-li příkaz COPY k zahrnutí struktury za položkou, která není položkou level-01 , ujistěte se, že struktura je násobkem 4-bajtů od začátku položky level-01 . Pokud to neuděláte, můžete snížit výkon aplikace.

Struktury jsou popsány v části Datové typy použité v rozhraní MQI. Popisy polí ve strukturách zobrazují názvy polí bez předpony. V programech COBOL, prefix názvů polí s názvem struktury následované pomlčkou, jak je uvedeno v deklaracích COBOL. Pole v souborech kopií struktury se tímto způsobem připojí jako předpona.

Názvy polí v deklaracích v souborech kopií struktury jsou velkými písmeny. Místo toho můžete použít malá i velká písmena. Například pole *StrucId* struktury MQGMO je zobrazeno jako MQGMO-STRUCID v deklaraci COBOL a v souboru kopie.

Struktury V-suffix jsou deklarovány s počátečními hodnotami pro všechna pole, takže je třeba nastavit pouze ta pole, kde se požadovaná hodnota liší od počáteční hodnoty.

Cování v C

Všimněte si informací v následujících sekcích při kódování programů WebSphere MQ v C.

- “Parametry volání MQI” na stránce 80
- “Parametry s nedefinovaným datovým typem” na stránce 80
- “Datové typy” na stránce 80
- “Manipulace s binárními řetězci” na stránce 80
- “Manipulace se znakovými řetězci” na stránce 81

Parametry volání MQI

Parametry, které jsou *pouze pro vstup* a typu MQHCONN, MQBOBJ, MQHMSG nebo MQLONG, jsou předávány hodnotou; pro všechny ostatní parametry je hodnota parametru předávána hodnotou parametru *address* .

Ne všechny parametry, které jsou předávány podle adresy, je třeba zadat pokaždé, když je vyvolána funkce. Není-li určitý parametr požadován, lze jako parametr při vyvolání funkce zadat ukazatel Null jako parametr na místo adresy dat parametru. Parametry, pro které je to možné, jsou identifikovány v popisech volání.

Jako hodnotu funkce se nevrací žádný parametr; v terminologii C to znamená, že všechny funkce vrací neobsazenou hodnotu.

Atributy funkce jsou definovány proměnnou makra MQENTRY; hodnota této proměnné makra závisí na daném prostředí.

Parametry s nedefinovaným datovým typem

Každý z funkcí MQGET, MQPUT a MQPUT1 má parametr *Buffer* , který má nedefinovaný datový typ. Tento parametr se používá k odesílání a přijímání dat zprávy aplikace.

Parametry tohoto řazení jsou zobrazeny v příkladech C jako pole MQBYTE. Parametry můžete deklarovat tímto způsobem, ale je obvykle výhodnější deklarovat je jako strukturu, která popisuje rozvržení dat ve zprávě. Parametr funkce je deklarovaný jako ukazatel na neobsazený, a proto lze jako parametr ve vyvolání funkce určit adresu jakýchkoli dat.

Datové typy

Všechny datové typy jsou definovány pomocí příkazu `typedef` .

Pro každý datový typ je také definován odpovídající datový typ ukazatele. Název datového typu ukazatele je název elementárních nebo strukturních datových typů s předponou písmenem P, která označuje ukazatel. Atributy ukazatele jsou definovány proměnnou makra MQPOINTER; hodnota této proměnné makra závisí na daném prostředí. Následující kód ukazuje, jak deklarovat datové typy ukazatele:

```
#define MQPOINTER                  /* depends on environment */
...
typedef MQLONG  MQPOINTER PMQLONG;  /* pointer to MQLONG */
typedef MQMD   MQPOINTER PMQMD;     /* pointer to MQMD */
```

Manipulace s binárními řetězci

Řetězce binárních dat jsou deklarovány jako jeden z datových typů MQBYTEn.

Kdykoli kopírujete, porovnávají nebo nastavují pole tohoto typu, použijte funkce C `memcpy`, `memcmp`nebo `memset`:

```
#include <string.h>
#include "cmqc.h"

MQMD MyMsgDesc;

memcpy(MyMsgDesc.MsgId,           /* set "MsgId" field to nulls    */
       MQMI_NONE,                 /* ...using named constant       */
       sizeof(MyMsgDesc.MsgId));

memset(MyMsgDesc.CorrelId,        /* set "CorrelId" field to nulls */
```

```
       0x00,                       /* ...using a different method   */
       sizeof(MQBYTE24));
```

Nepoužívejte řetězcové funkce strcpy, strcmp, strncpy nebo strncmp, protože tyto funkce nefungují správně s daty deklarovanými jako MQBYTE24.

Manipulace se znakovými řetězci

Když správce front vrátí do aplikace znaková data, správce front vždy vycpávku znaková data s mezerami do definované délky pole. Správce front nevrací řetězce ukončené hodnotou null, ale můžete je použít ve svém vstupu. Proto při kopírování, porovnání nebo zřetězení takových řetězců použijte řetězec funkcí strncpy, strncmp nebo strncat.

Nepoužívejte funkce řetězce, které vyžadují, aby řetězec byl ukončen znakem hex 00 (strcpy, strcmp a strcat). Also, do not use the function strlen to determine the length of the string; use instead the sizeof function to determine the length of the field.

Počáteční hodnoty pro struktury

Soubor začlenění <cmqc.h> definuje různé proměnné maker, které můžete použít k poskytnutí výchozích hodnot pro struktury při deklarování instancí těchto struktur. Tyto proměnné maker mají názvy ve tvaru MQxxx_DEFAULT, kde MQxxx představuje název struktury. Použijte je takto:

```
MQMD   MyMsgDesc = {MQMD_DEFAULT};
MQPMO  MyPutOpts = {MQPMO_DEFAULT};
```

Pro některá znaková pole rozhraní MQI definuje konkrétní platné hodnoty (například pro pole *StrucId* nebo pole *Format* v produktu MQMD). Pro každou z platných hodnot jsou k dispozici dvě proměnné makra:

- Jedna makroproměnná definuje hodnotu jako řetězec s délkou, s výjimkou implikované hodnoty null, která přesně odpovídá definované délce pole. Například symbol ¬ představuje prázdný znak:

  ```
  #define MQMD_STRUC_ID "MD¬¬"
  #define MQFMT_STRING "MQSTR¬¬¬"
  ```

  Použijte tento formulář s funkcemi memcpy a memcmp.
- Další proměnná makra definuje hodnotu jako pole typu char; název této proměnné makra je název řetězce tvořená řetězcem s příponou _ARRAY. Příklad:

  ```
  #define MQMD_STRUC_ID_ARRAY 'M','D','¬','¬'
  #define MQFMT_STRING_ARRAY 'M','Q','S','T','R','¬','¬','¬'
  ```

  Tento formulář použijte k inicializaci pole, když je instance struktury deklarována s hodnotami, které jsou odlišné od proměnné poskytnuté proměnnou makra MQMD_DEFAULT.

Počáteční hodnoty pro dynamické struktury

Když se požaduje proměnný počet instancí struktury, jsou instance obvykle vytvářeny v hlavní paměti získávané dynamicky pomocí funkcí calloc nebo malloc.

Chcete-li inicializovat pole v těchto strukturách, doporučuje se následující technika:

1. Deklarujte instanci struktury pomocí příslušné makro proměnné MQxxx_DEFAULT k inicializaci struktury. Tato instance se stane *modelem* pro jiné instance:

   ```
   MQMD ModelMsgDesc = {MQMD_DEFAULT};
                                     /* declare model instance */
   ```

Kódováním statických nebo automatických klíčových slov v deklaraci poskytnete instanci modelu statickou nebo dynamickou dobu životnosti podle potřeby.

2. K získání paměti pro dynamickou instanci struktury použijte funkce calloc nebo malloc:

```
PMQMD InstancePtr;
InstancePtr = malloc(sizeof(MQMD));
                                  /* get storage for dynamic instance */
```

3. Použijte funkci memcpy ke zkopírování instance modelu do dynamické instance:

```
memcpy(InstancePtr,&ModelMsgDesc,sizeof(MQMD));
                                  /* initialize dynamic instance */
```

Použít z C++

V případě programovacího jazyka C + + obsahují hlavičkové soubory následující další příkazy, které jsou zahrnuty pouze v případě použití kompilátoru C + +:

```
#ifdef __cplusplus
  extern "C" {
#endif

/* rest of header file */

#ifdef __cplusplus
  }
#endif
```

Kódování v jazyce COBOL

Všimněte si informací v následující sekci při kódování programů WebSphere MQ v jazyce COBOL.

Pojmenované konstanty

Názvy konstant jsou zobrazeny obsahující znak podtržítka (_) jako součást názvu. V COBOLu, musíte použít znak pomlčky (-) místo podtržítka. Konstanty, které mají hodnoty znakových řetězců, používají jako oddělovač řetězců znak apostrofu ('). Má-li kompilátor přijmout tento znak, použijte volbu kompilátoru APOST.

Kopírovaná soubor CMQV obsahuje deklarace pojmenovaných konstant s položkami level-10 . Chcete-li použít konstanty, deklarujte explicitně položku level-01 , pak použijte příkaz COPY ke kopírování v deklaracích konstant:

```
 WORKING-STORAGE SECTION.
 01  MQM-CONSTANTS.
     COPY CMQV.
```

Tato metoda však způsobí, že se konstanty zabírají v programu i v případě, že na ně není odkazováno. Pokud jsou konstanty zahrnuty do mnoha samostatných programů v rámci stejné jednotky spuštění, bude existovat více kopií konstant; to může mít za následek použití významného množství hlavní paměti. Tomuto problému se můžete vyhnout přidáním klauzule GLOBAL do deklarace level-01 :

```
* Declare a global structure to hold the constants
 01  MQM-CONSTANTS GLOBAL.
     COPY CMQV.
```

Toto alokuje paměť pouze pro *jednu* sadu konstant v rámci jednotky spuštění; konstanty se však mohou odvolávkou na *libovolný* program v rámci spuštěné jednotky, nejen program, který obsahuje deklaraci level-01 .

Zajištění zarovnání struktury

Měli byste dbát na to, aby struktury IBM WebSphere MQ , které jsou předávány ke spuštění na volání MQ , byly zarovnány na hranicích slova. Hranice slova je 4 bajty pro 32bitové procesy, 8 bajtů pro 64 bitové procesy a 16 bajtů pro 128bitové procesy (IBM i).

Kde je to možné, umístěte všechny struktury IBM WebSphere MQ dohromady tak, aby byly všechny zarovnány podél hranic.

Kódování v pTAL

Při kódování programů IBM WebSphere MQ v pTALsi povšimněte informací v následující části.

HP Integrity NonStop Server

Definování a inicializace struktur IBM WebSphere MQ

Definice struktury pTAL pro struktury IBM WebSphere MQ jsou poskytnuty s názvy, které končí na ^DEF. Například následující deklarace pTAL by byly kódovány tak, aby vytvořily strukturu deskriptoru IBM WebSphere MQ Message Descriptor (MQMD) a strukturu IBM WebSphere MQ Put Message Options (MQPMO).

```
STRUCT MYMD(MQMD^DEF);        ! Declare an MQMD structure
STRUCT MYPMO(MQPMO^DEF);      ! Declare an MQPMO structure
```

IBM WebSphere MQ poskytuje pTAL DEFINE s názvy, které končí na ^DEFAULT pro inicializaci IBM WebSphere MQ struktur s výchozími hodnotami. Následující příkazy pTAL jsou kódovány pro přiřazení výchozích hodnot deklarovaným strukturám MQMD a MQPMO:

```
MQMD^DEFAULT(MYMD);           ! Assign default values to an MQMD structure
MQPMO^DEFAULT(MYPMO);         ! Assign default values to an MQPMO structure
```

Můžete deklarovat a inicializovat jiné struktury IBM WebSphere MQ pomocí podobného kódu.

pTAL a CRE

Programy pTAL nemohou inicializovat prostředí Common Runtime Environment, a proto musí být použity s hlavní rutinou jazyka C nebo jazyka COBOL.

Ukázky pTAL , které jsou poskytovány s produktem IBM WebSphere MQ , používají rutinu mainline jazyka C s názvem AMQSPTM0.C

Parametry s datovým typem MQCHAR

Procedury MQGET, MQPUT a MQPUT1 mají každý parametr **Buffer** , který má datový typ MQCHAR .EXT . Tento parametr se používá k odeslání a přijetí dat zprávy aplikace.

Parametry tohoto řazení jsou zobrazeny v ukázkách pTAL jako pole řetězce. Tímto způsobem můžete deklarovat parametry, ale obvykle je výhodnější je deklarovat jako strukturu, která popisuje rozvržení dat ve zprávě. Parametr procedury je deklarován jako MQCHAR .EXT, ale adresu libovolných dat lze zadat jako parametr při vyvolání procedury.

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.

Kódování ve Visual Basicu

Při kódování programů WebSphere MQ ve Visual Basicu si všimněte informací v následující sekci.

Poznámka: Mimo prostředí .NET se úroveň podpory jazyka Visual Basic (VB) v produktu WebSphere MQ stabilizovala na úrovni V6.0 . Většina nových funkcí přidaných do produktu WebSphere MQ 7.0 nebo novější není k dispozici pro aplikace VB. Pokud programuji v VB.NET, použijte třídy produktu WebSphere MQ .NET. Další informace lze najít v tématu Použití rámce .NET.

Jazyk Visual Basic je podporován pouze v systému Windows.

Chcete-li se vyhnout nechtěnému překladu binárních dat mezi Visual Basic a WebSphere MQ, použijte definici MQBYTE místo MQSTRING. CMQB.BAS definuje několik nových typů MQBYTE, které jsou ekvivalentní definice typu C byte a používají je v rámci struktur WebSphere MQ . Například pro strukturu MQMD (deskriptor zprávy) je položka MsgId (identifikátor zprávy) definována jako MQBYTE24.

Visual Basic nemá datový typ ukazatele, takže odkazy na jiné datové struktury WebSphere MQ jsou odečteno ukazateli než ukazatelem. Deklarujte složenou strukturu skládající se ze dvou struktur komponent a určete složenou strukturu na volání. Podpora produktu WebSphere MQ pro Visual Basic poskytuje volání MQCONNXAny, aby bylo možné tuto možnost zpřístupnit a umožnit klientským aplikacím určit vlastnosti kanálu v rámci připojení klienta. Přijímá netypované struktury (MQCNOCD) místo typické struktury MQCNO.

Struktura MQCNOCD je složená struktura skládající se z objektu MQCNO a za ním příkaz MQCD. Tato struktura je deklarována v souboru záhlaví uživatelských procedur CMQXB. Pomocí rutiny MQCNOCD_DEFAULTS inicializujte strukturu MQCNOCD. K dispozici je ukázka volání MQCONNX (amqscnxb.vbp).

MQCONNXAny má stejné parametry jako MQCONNX, s výjimkou toho, že parametr *ConnectOpts* je deklarován jako typ Any, spíše než datový typ MQCNO. To umožňuje, aby funkce přijala buď strukturu MQCNO, nebo strukturu MQCNOCD. Tato funkce je deklarována v hlavním souboru záhlaví CMQB.

Objektový model produktu IBM WebSphere MQ

Model objektu IBM WebSphere MQ se skládá ze tříd, metod a vlastností. Tyto informace použijte k seznámení se s každým z těchto koncepcí.

Model objektu produktu IBM WebSphere MQ se skládá z následujících položek:

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

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

Tato témata popisují podrobné informace o jednotlivých modelech objektů WebSphere MQ :

- “Třídy” na stránce 84
- “odkazy na objekty” na stránce 85
- “Návratové kódy” na stránce 85

Třídy

Model objektu WebSphere MQ poskytuje následující základní sadu tříd.

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

MQQueueManager
Objekt třídy MQQueueManager představuje připojení ke správci front. Má metody Connect (), Disconnect (), Commit () a Backout () (ekvivalentní MQCONN nebo MQCONNX, MQDISC, MQCMIT

a MQBACK). Má vlastnosti odpovídající atributům správce front. Přístup k vlastnosti atributu správce front se implicitně připojuje ke správci front, pokud již není připojen. Destrojení objektu MQQueueManager se implicitně odpojí od správce front.

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

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

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

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

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

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

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

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

odkazy na objekty

V programu WebSphere MQ , který používá rozhraní MQI, vrací produkt WebSphere MQ k tomuto programu obslužné rutiny připojení a popisovače objektů.

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

Návratové kódy

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

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

Měla bych používat třídy IBM WebSphere MQ pro třídy Java nebo IBM WebSphere MQ pro platformu JMS?

Aplikace v jazyce Java může pro přístup k prostředkům produktu IBM WebSphere MQ používat třídy IBM WebSphere MQ pro třídy Java nebo IBM WebSphere MQ pro rozhraní JMS. Každý přístup má své výhody.

Třídy IBM WebSphere MQ pro prostředí Java zapouzdřují rozhraní MQI (Message Queue Interface), nativní rozhraní API produktu IBM WebSphere MQ a používá stejný objektový model jako jiná objektově orientovaná rozhraní, zatímco třídy produktu IBM WebSphere MQ pro platformu JMS (Java Message Service) implementují rozhraní JMS (Java Message Service) společnosti Sun.

Pokud jste obeznámeni s produktem IBM WebSphere MQ v jiných prostředích než Java, s použitím procedurálních nebo objektově orientovaných jazyků, můžete přenést své existující znalosti do prostředí Java pomocí tříd produktu IBM WebSphere MQ pro jazyk Java. Můžete také využít celou řadu funkcí produktu IBM WebSphere MQ, které nejsou všechny dostupné ve třídách produktu IBM WebSphere MQ pro platformu JMS.

Pokud nejste obeznámeni s produktem IBM WebSphere MQnebo pokud již máte zkušenosti s platformou JMS, může být snazší použít známé rozhraní API JMS pro přístup k prostředkům produktu IBM WebSphere MQ pomocí tříd produktu IBM WebSphere MQ pro platformu JMS. Platforma JMS je také integrální součástí platformy Java Platform, Enterprise Edition (Java EE). Aplikace Java EE mohou používat objekty typu message-driven bean (MDB) k asynchronnímu zpracování zpráv a objekty MDB mohou zpracovávat pouze zprávy JMS. Platforma JMS je také standardním mechanizmem pro interakci s asynchronními systémy zasílání zpráv Java EE , jako je například produkt IBM WebSphere MQ. Každý aplikační server, který je v souladu se standardem Java EE , musí obsahovat poskytovatele rozhraní JMS, a proto můžete použít JMS ke komunikaci mezi různými aplikačními servery nebo můžete aplikaci z jednoho poskytovatele rozhraní JMS portovat bez jakýchkoli změn v aplikaci.

Návrh aplikací produktu IBM WebSphere MQ

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

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

Typ aplikace

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

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

Kromě toho můžete také napsat své vlastní aplikace k automatizaci administrace produktu IBM WebSphere MQ. Další informace viz Úvod do administrativního rozhraní produktu WebSphere MQ (MQAI) a Automatizace administračních úloh .

Programovací jazyk

IBM WebSphere MQ podporuje řadu procedurálních a objektově orientovaných programovacích jazyků pro psaní aplikací. Další informace viz “Rozhodování o tom, jaký programovací jazyk použít” na stránce 75.

Aplikace pro více platforem

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

Používáte-li kód C, kód ANSI standard C. Používejte standardní funkce knihovny jazyka C a nikoli ekvivalentní funkce specifické pro platformu, a to i v případě, že funkce specifická pro danou

platformu je rychlejší nebo efektivnější. Výjimkou je, když je účinnost v kódu prvořadá, pokud byste měli zadat kód pro obě situace pomocí #ifdef. Příklad:

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

Typy front

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

Použití klastrů správců front

Možná budete chtít využít zjednodušené administrace systému a zvýšit dostupnost, rozšiřitelnost a vyrovnávání pracovní zátěže, které jsou možné, když používáte klastry. Další informace naleznete v tématu Klastry správců front .

Typy zpráv

Možná budete chtít použít datagramy pro jednoduché zprávy, ale požadovat zprávy (pro které očekáváte odpovědi) pro jiné situace. Možná budete chtít některé z vašich zpráv přiřadit jiné priority. Další informace o návrhu zpráv viz “Návrh zpráv” na stránce 88.

Použití publikování/odběru nebo zasílání zpráv mezi dvěma body

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

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

Řízení vašich programů IBM WebSphere MQ

Možná budete chtít spustit některé programy automaticky nebo aby programy čekaly, dokud určitá zpráva nepřijde do fronty (pomocí funkce IBM WebSphere MQ *triggering* , viz “Spuštění aplikací produktu IBM WebSphere MQ pomocí spouštěčů” na stránce 316). Eventuálně můžete chtít spustit další instanci aplikace, když se zprávy ve frontě nezpracovávají dostatečně rychle (pomocí funkce IBM WebSphere MQ *instrumentation events* , jak je popsáno v části Události instrumentace).

Spuštění vaší aplikace na klientovi IBM WebSphere MQ

V prostředí klienta je podporována úplná rozhraní MQI a tato operace umožňuje téměř všechny aplikace IBM WebSphere MQ , které mají být předány ke spuštění na klientovi IBM WebSphere MQ MQI. Link the application on the IBM WebSphere MQ MQI client to the MQIC library, rather than to the MQI library.

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

Další informace viz “Spuštění aplikací v prostředí klienta IBM WebSphere MQ MQI” na stránce 344.

Výkon aplikace

Konstrukční rozhodnutí mohou mít dopad na výkon aplikací, a to pro návrhy na zvýšení výkonu aplikací produktu IBM WebSphere MQ , viz “Návrh a výkon aplikací” na stránce 89.

Rozšířené metody IBM WebSphere MQ

Pro pokročilejší aplikace může být vhodné použít některé rozšířené metody IBM WebSphere MQ , jako např. korelace odpovědí a generování a odesílání informací o kontextu produktu IBM WebSphere MQ . Další informace viz téma “Rozšířené metody IBM WebSphere MQ” na stránce 91.

Zabezpečení vašich dat a správa jeho integrity

Můžete použít informace o kontextu předané spolu se zprávou a otestovat, zda byla zpráva odeslána z přípustného zdroje. Zařízení syncpointing, kterou poskytuje produkt IBM WebSphere MQ nebo váš operační systém, můžete použít k zajištění konzistence dat s ostatními prostředky (další podrobnosti viz “Potvrzení a zálohování jednotek práce” na stránce 310). Funkci *perzistence* zpráv produktu IBM WebSphere MQ můžete použít k zajištění doručení důležitých zpráv.

Testování aplikací produktu IBM WebSphere MQ

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

Práce s výjimkami a chybami

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

Související pojmy

Technický přehled produktu IBM WebSphere MQ

“Koncepty vývoje aplikací” na stránce 7
K zápisu aplikací IBM WebSphere MQ můžete použít volbu procedurálních nebo objektově orientovaných jazyků. Informace o konceptech produktu IBM WebSphere MQ , které jsou užitečné pro vývojáře aplikací, naleznete v odkazech v tomto tématu.

“Psaní front aplikace” na stránce 187
Prostřednictvím těchto informací můžete získat informace o vytváření aplikací pro řazení do front, připojování a odpojování od správce front, publikování a odběru a otevírání a zavírání objektů.

“Tvorba klientských aplikací” na stránce 337
Co potřebujete vědět, chcete-li zapisovat klientské aplikace do produktu WebSphere MQ.

“Použití .NET” na stránce 540
Třídy WebSphere MQ pro prostředí .NET umožňují programu napsaným v programovacím rámci .NET pro připojení k produktu WebSphere MQ jako klienta WebSphere MQ MQI nebo k přímému připojení k serveru WebSphere MQ .

“Použití C++” na stránce 607
Produkt WebSphere MQ poskytuje třídy C + + ekvivalentní objekty WebSphere MQ a některé další třídy ekvivalentní k datovým typům pole. Poskytuje řadu funkcí, které nejsou prostřednictvím rozhraní MQI k dispozici.

“Použití tříd produktu WebSphere MQ pro službu JMS” na stránce 692
Třídy WebSphere MQ pro platformy JMS (WebSphere MQ Classes for JMS) jsou poskytovatelem rozhraní JMS dodávaným s produktem WebSphere MQ. Kromě implementace rozhraní definovaných v balíku javax.jms produkt WebSphere MQ Classes for JMS poskytuje dvě sady rozšíření rozhraní JMS API.

“Použití tříd produktu WebSphere MQ pro prostředí Java” na stránce 631
Třídy WebSphere MQ pro prostředí Java umožňují používat produkt WebSphere MQ v prostředí Java. A Java application can use either WebSphere MQ classes for Java or WebSphere MQ classes for JMS to access WebSphere MQ resources.

“Použití rozhraní modelu objektu Component Interface (WebSphere MQ Automation Classes for ActiveX)” na stránce 1000
Produkt WebSphere MQ Automation Classes for ActiveX (MQAX) jsou komponenty ActiveX , které poskytují třídy, které můžete použít ve své aplikaci k přístupu k produktu WebSphere MQ.

Návrh zpráv

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

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

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

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

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

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

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

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

Vliv restartování správce front na zprávy
Správce front zachová všechny trvalé zprávy a po restartování bude v případě potřeby zotavována ze souborů protokolu produktu WebSphere MQ . Netrvalé zprávy a dočasné dynamické fronty nejsou zachovány. Všechny zprávy, které nechcete vyřadit, musí být definovány jako trvalé, když jsou vytvořeny. Při zápisu aplikace pro produkt WebSphere MQ pro produkt Windows nebo WebSphere MQ v systému UNIX and Linux se ujistěte, že víte, jak byl systém nastaven pro přidělení souboru protokolu, aby se snížilo riziko návrhu aplikace, která bude spuštěna na omezení souboru protokolu.

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

Objem přijímacích front
Pokud může být třeba vložit zprávu do několika front, můžete použít distribuční seznamnebo publikovat v rámci tématu.

Návrh a výkon aplikací

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

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

- Navrhněte svou aplikaci tak, aby zpracování probíhá souběžně s časem přemýšlení uživatele:
  - Zobrazí panel a umožní uživateli začít psát, dokud se aplikace stále inicializuje.
  - Získejte data, která potřebujete paralelně, z různých serverů.
- Ponechejte připojení a fronty otevřené, pokud je budete opakovaně používat místo opakovaného otevírání a zavírání, připojování a odpojování.
- Avšak serverová aplikace, která vkládá pouze jednu zprávu, by měla použít MQPUT1.
- Správci front jsou optimalizovány pro zprávy v rozsahu 4 KB až 100 kB. Velmi velké zprávy jsou neefektivní; je pravděpodobně lepší posílat 100 zpráv o velikosti 1 MB, každý než 100 MB zprávy. Velmi

malé zprávy jsou také neefektivní. Správce front provádí stejné množství práce pro jednobajtovou zprávu jako pro zprávu o velikosti 4 kB.

- Udržujte své zprávy v rámci pracovní jednotky, aby mohly být potvrzeny nebo zálohovány současně.
- Použijte dočasnou volbu pro zprávy, které není třeba obnovit.
- Potřebujete-li odeslat zprávu na určitý počet cílových front, zvažte použití distribučního seznamu.

Vliv délky zprávy

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

Vliv perzistence zpráv

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

Hledání konkrétní zprávy

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

Fronty, které obsahují zprávy o různých délkách

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

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

Frekvence synchronizačních bodů

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

Použití volání MQPUT1

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

Počet používaných podprocesů

V případě produktu WebSphere MQ for Windowsmůže aplikace vyžadovat velký počet podprocesů. Každému procesu správce front je přidělen maximální povolený počet podprocesů aplikací.

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

Rozšířené metody IBM WebSphere MQ

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

Čekání na zprávy

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

- Čeká se na doručení zprávy, nebo uplynutí zadaného časového intervalu (viz “Čekání na zprávy” na stránce 256).
- Vytvoření uživatelské procedury zpětného volání, která má být řízena při doručení zprávy; viz “Asynchronní spotřeba zpráv produktu IBM WebSphere MQ” na stránce 32.
- Pravidelná volání do fronty za účelem zjištění, zda byla doručena zpráva (*polling*). Tato situace se obvykle nedoporučuje, protože může mít vliv na výkon.

Korelace odpovědí

Když v aplikacích WebSphere MQ program obdrží zprávu, která požaduje její provedení některé práce, program obvykle odešle žadateli jednu nebo více zpráv odpovědí.

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

Nastavení a použití kontextových informací

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

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

Další informace o používání a nastavení kontextových informací viz “kontext zprávy” na stránce 37.

Automatické spuštění programů produktu WebSphere MQ

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

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

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

Další informace o spouštění naleznete v tématu “Spuštění aplikací produktu IBM WebSphere MQ pomocí spouštěčů” na stránce 316. Spouštěcí impuls je pouze jedním ze způsobů, jak spustit program automaticky. Můžete například spustit program automaticky v časovači pomocí zařízení jiného typu nežWebSphere MQ .

WebSphere MQ může definovat objekty služeb pro spuštění programů WebSphere MQ při spuštění správce front; viz Objekty služeb.

Generování sestav WebSphere MQ

V aplikaci můžete požadovat následující sestavy:

- Sestavy výjimek

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

Ty jsou popsány v části “Hlášení zpráv” na stránce 11.

Klastry a afinity zpráv

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

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

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

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

Další informace najdete v tématu Práce s spřízněnostmi zpráv .

Ukázka programů WebSphere MQ

Tato kolekce témat slouží k získání informací o ukázkových programech WebSphere MQ na různých platformách.

- “Ukázkové programy pro distribuované platformy” na stránce 93

Související pojmy

“Koncepty vývoje aplikací” na stránce 7
K zápisu aplikací IBM WebSphere MQ můžete použít volbu procedurálních nebo objektově orientovaných jazyků. Informace o konceptech produktu IBM WebSphere MQ , které jsou užitečné pro vývojáře aplikací, naleznete v odkazech v tomto tématu.

“Rozhodování o tom, jaký programovací jazyk použít” na stránce 75
Tyto informace použijte k vyhledání informací o programovacích jazycích a rámcích podporovaných produktem IBM WebSphere MQa o některých aspektech jejich použití.

“Návrh aplikací produktu IBM WebSphere MQ” na stránce 86
Když se rozhodnete, jak aplikace mohou využívat výhody platforem a prostředí, které máte k dispozici, musíte se rozhodnout, jak budete používat funkce nabízené produktem WebSphere MQ.

“Psaní front aplikace” na stránce 187
Prostřednictvím těchto informací můžete získat informace o vytváření aplikací pro řazení do front, připojování a odpojování od správce front, publikování a odběru a otevírání a zavírání objektů.

“Tvorba klientských aplikací” na stránce 337
Co potřebujete vědět, chcete-li zapisovat klientské aplikace do produktu WebSphere MQ.

“Použití webových služeb v produktu WebSphere MQ” na stránce 913
Můžete vyvíjet aplikace produktu IBM WebSphere MQ pro webové služby pomocí přenosu IBM WebSphere MQ pro protokol SOAP nebo mostu produktu IBM WebSphere MQ pro protokol HTTP.

“Zapisování aplikací typu publikování/odběr” na stránce 266
Začněte zapisovat do aplikací WebSphere MQ publish/subscribe.

“Sestavení aplikace IBM WebSphere MQ” na stránce 412
Tyto informace použijte k seznámení se s aplikací produktu IBM WebSphere MQ na různých platformách.

“Obsluha chyb programu” na stránce 529
Tyto informace vysvětlují chyby související s voláními MQI MQI buď při volání, nebo při doručení její zprávy do konečného cíle.

Ukázkové programy pro distribuované platformy

Toto téma popisuje vzorové programy dodávané s produktem IBM WebSphere MQ, napsané v jazycích C a COBOL. Ukázky demonstrují typická použití rozhraní MQI (Message Queue Interface).

Ukázky nejsou určeny k demonstraci obecných programovacích technik, takže se vynechává některá kontrola chyb, kterou byste mohli chtít zahrnout do produkčního programu. Tyto ukázky jsou však vhodné pro použití jako základ pro vlastní programy front zpráv.

Zdrojový kód pro všechny ukázky je dodáván spolu s produktem; tento zdroj zahrnuje komentáře, které vysvětlují techniky front zpráv demonstrovány v programech.

Ukázkové programy C + +: Viz Použití C++ pro popis ukázkových programů, které jsou k dispozici v C + +.

Názvy ukázek začínají předponou amq. Čtvrtý znak označuje programovací jazyk a v případě potřeby kompilátor.

s	Jazyk C
0	Jazyk COBOL na obou kompilátorech IBM a Micro Focus
i	Jazyk COBOL pouze na kompilátorech IBM
m	Jazyk COBOL pouze na kompilátorech Micro Focus

Osmý znak spustitelného souboru označuje, zda je ukázka spuštěna v režimu lokální vazby nebo v režimu klienta. Pokud žádný osmý znak neobsahuje, je ukázka spuštěna v režimu lokálních vazeb. Je-li osmý znak ' c', ukázka se spustí v režimu klienta. Chcete-li nastavit správce front tak, aby přijímal klientská připojení, prohlédněte si téma “Příprava a spuštění ukázkových programů” na stránce 104 , kde získáte podrobnosti.

Chcete-li zjistit více o ukázkových programech, použijte následující odkazy:

- “Vlastnosti demonstrovány v ukázkových programech” na stránce 94
- “Ukázkové programy publikování/odběru” na stránce 129
- “Ukázkové programy Put” na stránce 135
- “Ukázkový program Distribuční seznam” na stránce 122
- “Ukázkové programy Procházet” na stránce 111
- “Ukázkový program prohlížeče” na stránce 112
- “Ukázkové programy Get” na stránce 123
- “Ukázkové programy Referenční zprávy” na stránce 136
- “Ukázkové programy požadavku” na stránce 141
- “Ukázkové programy pro zjišťování” na stránce 128
- “The Inquire Properties of a Message Handle sample program” na stránce 129
- “Ukázkové programy Set” na stránce 145
- “Ukázkové programy Echo” na stránce 123
- “Ukázkový program pro převod dat” na stránce 114
- “Spouštěcí ukázkové programy” na stránce 148
- “Ukázkový program Asynchronous Put” na stránce 110
- “Ukázky koordinace databází” na stránce 115
- “Ukázka transakce CICS” na stránce 113
- “Ukázky TUXEDO” na stránce 149

- “Ukázka obslužné rutiny fronty nedoručených zpráv” na stránce 121
- “Ukázkový program Connect” na stránce 113
- “Ukázkový program uživatelské procedury rozhraní API” na stránce 108
- “Použití uživatelské procedury zabezpečení SSPI v systémech Windows” na stránce 163
- “Spuštění ukázek pomocí vzdálených front” na stránce 164
- “Ukázkový program pro monitorování front klastru (AMQSCLM)” na stránce 164
- “Ukázkový program pro vyhledávání koncového bodu připojení (CEPL)” na stránce 173

Vlastnosti demonstrovány v ukázkových programech

Kolekce tabulek, které zobrazují techniky, které demonstrují ukázkové programy produktu WebSphere MQ .

Všechny ukázky otevírají a zavírají fronty pomocí volání MQOPEN a MQCLOSE, takže tyto techniky nejsou v tabulkách uvedeny odděleně. Podívejte se do záhlaví, které zahrnuje platformu, o kterou máte zájem.

Ukázky pro systémy UNIX and Linux

Toto téma obsahuje techniky, které demonstrují ukázkové programy pro produkt WebSphere MQ v systémech UNIX and Linux .

Informace o tom, kde jsou uloženy ukázkové programy pro produkt WebSphere MQ v systémech UNIX a Linux , naleznete v tématu “Příprava a spuštění ukázkových programů v systémech UNIX” na stránce 106 .

Tabulka 14 na stránce 94 Tabulka uvádí, které zdrojové soubory C a COBOL jsou poskytovány, a zda je zahrnut server nebo klientský spustitelný soubor.

Tabulka 14. WebSphere MQ v ukázkových programech produktu UNIX and Linux demonstrujících použití MQI (C a COBOL)

Technika	**C (zdroj) (“1” na stránce 96)**	**COBOL (zdroj) (“2” na stránce 96)**	**Server (spustitelný soubor C)**	**Klient (spustitelný soubor C) (“3” na stránce 96)**
Použití rozhraní pro publikování/odběr	amqspuba amqssuba amqssbxa	bez vzorku	amqspub amqssub amqssbx	bez vzorku
Vložení zpráv pomocí volání MQPUT	amqsput0	amq0put0	amqsput	amqsputc
Vložení jedné zprávy pomocí volání MQPUT1	amqsinqa amqsecha	amqminqx amqmechx amqiinqx amqiechx	amqsinq amqsech	amqsechc
Vložení zpráv do distribučního seznamu (“4” na stránce 96)	amqsptl0	amq0ptl0.cbl	amqsptl	amqsptlc
Odpověď na zprávu požadavku	amqsinqa	amqminqx amqiinqx	amqsinq	bez vzorku
Získávání zpráv (bez čekání)	amqsgbr0	amq0gbr0	amqsgbr	bez vzorku
Získávání zpráv (čekání s časovým limitem)	amqsget0	amq0get0	amqsget	amqsgetc
Získávání zpráv (neomezená doba čekání)	amqstrg0	bez vzorku	amqstrg	amqstrgc
Získávání zpráv (s převodem dat)	amqsecha	bez vzorku	amqsech	bez vzorku

Tabulka 14. WebSphere MQ v ukázkových programech produktu UNIX and Linux demonstrujících použití MQI (C a COBOL) (pokračování)

Technika	**C (zdroj) (“1” na stránce 96)**	**COBOL (zdroj) (“2” na stránce 96)**	**Server (spustitelný soubor C)**	**Klient (spustitelný soubor C) (“3” na stránce 96)**
Vložení odkazové zprávy do fronty (“4” na stránce 96)	amqsprma	bez vzorku	amqsprm	amqsprc
Získání referenčních zpráv z fronty (“4” na stránce 96)	amqsgrma	bez vzorku	amqsgrm	amqsgrmc
Uživatelská procedura kanálu zpráv odkazu (“4” na stránce 96)	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	amqminqx amqiinqx	amqsinq	amqsinqc
Použití výlučné vstupní fronty	amqstrg0	amq0req0	amqstrg	amqstrgc
Použití volání MQINQ	amqsinqa	amqminqx amqiinqx	amqsinq	bez vzorku
Použití volání MQSET	amqsseta	amqmsetx amqisetx	amqsset	amqssetc
Použití fronty pro odpověď	amqsreq0	amq0req0	amqsreq	amqsreqc
Vyžádání výjimek zpráv	amqsreq0	amq0req0	amqsreq	bez vzorku
Přijímání ořízнuté zprávy	amqsgbr0	amq0gbr0	amqsgbr	bez vzorku
Použití vyřešeného názvu fronty	amqsgbr0	amq0gbr0	amqsgbr	bez vzorku
Spouštění procesu	amqstrg0	bez vzorku	amqstrg	amqstrgc
Použití konverze dat	(“5” na stránce 96)	bez vzorku	bez vzorku	bez vzorku
WebSphere MQ (koordinující XA-vyhovující správce databází) přístup k jedné databázi pomocí SQL	amqsxas0.sqc DB2 amqsxas0.ec Informix	amq0xas0.sqb	bez vzorku	bez vzorku
WebSphere MQ (koordinující XA-kompatibilní správce databází) přístup k dvěma databázím pomocí jazyka SQL	amqsxag0.c amqsxab0.sqc amqsxaf0.sqc	amq0xag0.cbl amq0xab0.sqb amq0xaf0.sqb	bez vzorku	bez vzorku
Transakce CICS (“6” na stránce 96)	amqscic0.ccs	bez vzorku	amqscic0	bez vzorku
Transakce Encina (“4” na stránce 96)	amqsxae0	bez vzorku	amqsxae0	bez vzorku
Transakce TUXEDO pro vložení zpráv (“7” na stránce 96)	amqstxpx	bez vzorku	bez vzorku	bez vzorku
Transakce TUXEDO pro získání zpráv (“7” na stránce 96)	amqstxgx	bez vzorku	bez vzorku	bez vzorku

Tabulka 14. WebSphere MQ v ukázkových programech produktu UNIX and Linux demonstrujících použití MQI (C a COBOL) (pokračování)

Technika	**C (zdroj) (“1” na stránce 96)**	**COBOL (zdroj) (“2” na stránce 96)**	**Server (spustitelný soubor C)**	**Klient (spustitelný soubor C) (“3” na stránce 96)**
Server pro TUXEDO (“7” na stránce 96)	amqstxsx	bez vzorku	bez vzorku	bez vzorku
obslužná rutina fronty nedoručených zpráv	Adresář `./tools/c/Samples/dlq` (“8” na stránce 96)	bez vzorku	amqsdlq	bez vzorku
odeslání zprávy z klienta MQI	bez vzorku	bez vzorku	bez vzorku	amqsputc
Z klienta MQI se získáním zprávy	bez vzorku	bez vzorku	bez vzorku	amqsgetc
Připojení ke správci front pomocí příkazu MQCONNX	amqscnxc	bez vzorku	bez vzorku	amqscnxc
Použití ukončení rozhraní API	amqsaxe0	bez vzorku	amqsaxe	bez vzorku
Ukončení vyrovnávání pracovní zátěže klastru	amqswlm0	bez vzorku	amqswlm	bez vzorku
Asynchronně vkládání zpráv a získání stavu pomocí volání MQSTAT	amqsapt0	bez vzorku	amqsap	amqsaptc
Opakovaně připojitelní klienti	amqsphak amqsgak amqsmhak	bez vzorku	nelze použít	amqsphak amqsgak amqsmhak
Použití spotřebitelů zpráv k asynchronnímu příjmu zpráv z více front	amqscbf0	bez vzorku	amqscbf	amqscbfc
Určení informací o připojení SSL/TLS pro MQCONNX	amqssslc	bez vzorku	nelze použít	amqssslc

Notes:

1. Spustitelná verze ukázek klienta WebSphere MQ MQI sdílí stejný zdroj jako ukázky, které se spouštějí v prostředí serveru.
2. Kompilační programy začínající 'amqm' se kompilátorem Micro Focus COBOL, ty začínající 'amqi' s kompilátorem IBM COBOL a s těmi začínajícími 'amq0'.
3. Spustitelné verze ukázek klienta WebSphere MQ MQI nejsou k dispozici v produktu WebSphere MQ for HP-UX.
4. Podporováno na produktech WebSphere MQ pro AIX, WebSphere MQ for HP-UXa WebSphere MQ pouze pro Solaris.
5. V produktu WebSphere MQ for AIX, WebSphere MQ for HP-UXa WebSphere MQ for Solaris se tento program nazývá `amqsvfc0.c` .
6. CICS je podporován pouze WebSphere MQ for AIX a WebSphere MQ pouze pro HP-UX .
7. Funkce TUXEDO není podporována produktem WebSphere MQ for Linux v systému System p.
8. Zdroj pro popisovač fronty nedoručených zpráv se skládá z několika souborů a je uveden v samostatném adresáři.

Podrobné informace o podpoře pro systémy UNIX and Linux jsou k dispozici na stránce s požadavky na systémy WebSphere MQ na adrese Systémové požadavky pro produkt IBM WebSphere MQ.

Ukázky pro klienta IBM WebSphere MQ for HP Integrity NonStop Server

Toto téma zobrazuje techniky demonstrovány ukázkovými programy pro klienta IBM WebSphere MQ na systémech HP Integrity NonStop Server .

Tabulka 15 na stránce 97Tabulka uvádí zdrojové vzorové programy C, COBOL a pTAL , které jsou k dispozici.

Tabulka 15. IBM WebSphere MQ v ukázkovém programu HP Integrity NonStop Server demonstrující použití jazyků C, COBOL a pTAL

Technika	**C**				**COBOL**		**pTAL**	
	OSS (Zdroj)	OSS (spustitelný soubor)	Guardian (zdroj)	Guardian (spustitelný soubor)	OSS (Zdroj)	Guardian (zdroj)	OSS (Zdroj)	Guardian (zdroj)
Použití rozhraní pro publikování/ odběr	amqspuba.c amqssbxa.c amqssuba.c amqspse0.c	amqspubc amqssbxc amqssubc	MQSPUBC MQSSBXC MQSSUBC	AMQSPUBC AMQSSBXC AMQSSUBC	amq0pub0.cbl amq0sub0.cbl	MQSPUBL MQSSUBL	amqtpub0.tal amqtsub0.tal	MQSPUBT MQSSUBT
Vložení zpráv pomocí volání MQPUT	amqsput0.c	amqsputc	MQSPUTC	amqsputc	amq0put0.cbl	MQSPUTL.	amqtput0.tal	MQSPUTT.
Vložení jedné zprávy pomocí volání MQPUT1	amqsecha.c	amqsechc	MQSECHC	AMQSECHC			amqtech0.tal	MQSECHT
Vložení zpráv do distribučního seznamu	amqsptl0.c	amqsptlc	MQSPTLC	AMQSPTLCK	amq0ptl0.cbl	MQSPTLL		
Odpověď na zprávu požadavku	amqsinqa.c	amqsinqc	MQSINQC	AMQSINQC				
Získávání zpráv (bez čekání)	amqsgbr0.c	amqsgbrc	MQSGBRC	AMQSGBRC	amq0gbr0.cbl	MQSGBRL		

Tabulka 15. IBM WebSphere MQ v ukázkovém programu HP Integrity NonStop Server demonstrující použití jazyků C, COBOL a pTAL (pokračování)

Technika	**C**				**COBOL**		**pTAL**	
Získávání zpráv (čekání s časovým limitem)	amqsget0.c	amqsgetc	MQSGETC.	AMQSGETC.	amq0get0.cbl	MQSGETL.	amqtget0.tal	MQSGETT.
Získávání zpráv (neomezená doba čekání)	amqstrg0.c	amqstrgc	MQSTRGC	AMQSTRGC				
Získávání zpráv (s převodem dat)	amqsecha.c	amqsechc	MQSECHC	AMQSECHC				
Vložení odkazové zprávy do fronty	amqsprma.c	amqsprc	MQSPRC	AMQSPRC				
Získání referenčních zpráv z fronty	amqsgrma.c	amqsgrmc	MQSGRMC	AMQGRMC				
Referenční uživatelská procedura kanálu zpráv	amqsqrma.c amqsxrma.c		MQSQRMC MQSXRMC					
Procházení prvních 20 znaků zprávy	amqsgbr0.c	amqsgbrc	MQSGBRC	AMQSGBRC	amq0gbr0.cbl	MQSGBRL		
Procházení kompletních zpráv	amqsbcg0.c	amqsbcgc	MQSBCGC	AMQSBCGC				
Použití sdílené vstupní fronty	amqsinqa.c	amqsinqc	MQSINQC	MQSINQC				

Tabulka 15. IBM WebSphere MQ v ukázkovém programu HP Integrity NonStop Server demonstrující použití jazyků C, COBOL a pTAL (pokračování)

Technika	**C**				**COBOL**		**pTAL**	
Použití výlučné vstupní fronty	amqstrg0.c	amqstrgc	MQSTRGC	AMQSTRGC				
Použití volání MQINQ	amqsinqa.c	amqsinqc	MQSINQC	AMQSINQC				
Použití volání MQSET	amqsseta.c	amqssetc	FUNKCE MQSSETC	AMQSSETC				
Použití fronty pro odpověď	amqsreq0.c	amqsreqc	MQSREQC	AMQSREQCUS	amq0req0.cbl	MQSREQL		
Vyžádání výjimek zpráv	amqsreq0.c	amqsreqc	MQSREQC	AMQSREQCUS	amq0req0.cbl	MQSREQL		
Přijímání oříznuté zprávy	amqsgbr0.c	amqsgbrc	MQSGBRC	AMQSGBRC	amq0gbr0.cbl	MQSGBRL		
Použití vyřešeného názvu fronty	amqsgbr0.c	amqsgbrc	MQSGBRC	AMQSGBRC	amq0gbr0.cbl	MQSGBRL		
Spouštění procesu	amqstrg0.c	amqstrgc	MQSTRGC	AMQSTRGC				
Použití konverze dat	amqsvfc0.c							
Obslužná rutina fronty nedoručených zpráv (1)	Adresář ./samp/dlq							
Připojení ke správci front pomocí příkazu MQCONNX	amqscnxc.c	amqscnxc	MQSCNXC					

Tabulka 15. IBM WebSphere MQ v ukázkovém programu HP Integrity NonStop Server demonstrující použití jazyků C, COBOL a pTAL (pokračování)

Technika	**C**				**COBOL**		**pTAL**	
Použití ukončení rozhraní API	amqsaxe0.c amqsaem0.c							
Ukončení vyrovnávání pracovní zátěže klastru	amqswlm0.c		MQSWLMC					
Monitor fronty klastru	amqsclma.c							
Asynchronně vkládání zpráv a získání stavu pomocí volání MQSTAT	amqsapt0.c	amqsaptc	MQSAPTC	MQSAPTC				
Opakovaně připojitelní klienti	amqsghac.c amqsmhac.c amqsphac.c	amqsghac amqsmhac amqsphac	MQSGAC MQSMHAC MQSPHAC MQSFHAC MQSPHAC MQSFHAC	AMQSGHAC AMQSMHAC AMQSPHAC AMQSFHAC				
Použití spotřebitelů zpráv k asynchronnímu příjmu zpráv z více front	amqscbf0.c	amqscbfc						
Určení informací o připojení SSL/TLS pro MQCONNX	amqsssslc.c	amqsssslc	MQSSSLC	AMQSSSLC				

Tabulka 15. IBM WebSphere MQ v ukázkovém programu HP Integrity NonStop Server demonstrující použití jazyků C, COBOL a pTAL (pokračování)

Technika	**C**				**COBOL**		**pTAL**	
Trasování aktivity	amqsact0.c	amqsactc	MQSACTC	AMQSACTC				
Vlastnosti zprávy	amqsiqma.c amqsstma.c	amqsiqmc amqsstmc	MQSIQMC MQSSTMC	AMQSIQMC AMQSSTMC				
Příkazový server	amqsstop.c		MQSSTOC.					
Protokolovat události	amqslog0.c	amqslogc	MQSLOGC	AMQSLOGC				
Účetnictví	amqsmon0.c	amqsmonc	MQSMONC	AMQSMONC				
rozhraní administrace	amqsaicq.c amqsaiem.c amqsailq.c							
Příklad hlavní funkce jazyka C pro vyvolání pTAL			FUNKCE MQSPTMC					

Notes:

1. Zdroj pro popisovač fronty nedoručených zpráv se skládá z několika souborů a je uveden v samostatném adresáři.
2. Informace o vývoji aplikací pro klienta IBM WebSphere MQ na platformě HP Integrity NonStop Server naleznete v následujících tématech:
   - “Vytváření vaší aplikace v systému HP Integrity NonStop Server” na stránce 419
     - “Příprava programů jazyka C v produktu HP Integrity NonStop Server” na stránce 421
     - “Příprava programů COBOL” na stránce 422
     - “Příprava programů pTAL” na stránce 423

Ukázky pro produkt IBM WebSphere MQ for Windows

Toto téma obsahuje techniky demonstrovány ukázkovými programy pro produkt IBM WebSphere MQ for Windows.

Tabulka 16 na stránce 102Tabulka uvádí, které zdrojové soubory C a COBOL jsou poskytovány, a zda je zahrnut server nebo klientský spustitelný soubor.

Tabulka 16. Ukázkové programy produktu IBM WebSphere MQ for Windows demonstrují použití rozhraní MQI (C a COBOL)

Technika	**C (zdroj)**	**COBOL (zdroj)**	**Server (spustitelný soubor C)**	**Klient (spustitelný soubor C)**
Použití rozhraní pro publikování/odběr	amqspuba amqssuba amqssbxa	bez vzorku	amqspub amqssub amqssbx	bez vzorku
Vložení zpráv pomocí volání MQPUT	amqsput0	amq0put0	amqsput	amqsputc
Vložení jedné zprávy pomocí volání MQPUT1	amqsinqa amqsecha	amqminq2 amqmech2 amqiinq2 amqiech2	amqsinq amqsech	amqsinqc amqsechc
Vložení zpráv do distribučního seznamu	amqsptl0	amq0ptl0.cbl	amqsptl	amqsptlc
Odpověď na zprávu požadavku	amqsinqa	amqminq2 amqiinq2	amqsinq	amqsinqc
Získávání zpráv (bez čekání)	amqsgbr0	amq0gbr0	amqsgbr	amqsgbrc
Získávání zpráv (čekání s časovým limitem)	amqsget0	amq0get0	amqsget	amqsgetc
Získávání zpráv (neomezená doba čekání)	amqstrg0	bez vzorku	amqstrg	amqstrgc
Získávání zpráv (s převodem dat)	amqsecha	bez vzorku	amqsech	amqsechc
Vložení odkazové zprávy do fronty	amqsprma	bez vzorku	amqsprm	amqsprc
Získání referenčních zpráv z fronty	amqsgrma	bez vzorku	amqsgrm	amqsgrmc
Referenční uživatelská procedura kanálu zpráv	amqsqrma amqsxrma	bez vzorku	amqsxrm	bez vzorku
Procházení prvních 20 znaků zprávy	amqsgbr0	amq0gbr0	amqsgbr	amqsgbrc
Procházení kompletních zpráv	amqsbcg0	bez vzorku	amqsbcg	amqsbcgc
Použití sdílené vstupní fronty	amqsinqa	amqminq2 amqiinq2	amqsinq	amqsinqc
Použití výlučné vstupní fronty	amqstrg0	amq0req0	amqstrg	amqstrgc
Použití volání MQINQ	amqsinqa	amqminq2 amqiinq2	amqsinq	amqsinqc
Použití volání MQSET	amqsseta	amqmset2 amqiset2	amqsset	amqssetc
Použití volání MQINQMP	amqsiqma	bez vzorku	bez vzorku	bez vzorku
Použití fronty pro odpověď	amqsreq0	amq0req0	amqsreq	amqsreqc
Vyžádání výjimek zpráv	amqsreq0	amq0req0	amqsreq	amqsreqc
Přijímání oříznuté zprávy	amqsgbr0	amq0gbr0	amqsgbr	amqsgbrc
Použití vyřešeného názvu fronty	amqsgbr0	amq0gbr0	amqsgbr	amqsgbrc
Spouštění procesu	amqstrg0	bez vzorku	amqstrg	amqstrgc
Použití konverze dat	amqsvfc0	bez vzorku	bez vzorku	bez vzorku

Tabulka 16. Ukázkové programy produktu IBM WebSphere MQ for Windows demonstrují použití rozhraní MQI (C a COBOL) (pokračování)

Technika	**C (zdroj)**	**COBOL (zdroj)**	**Server (spustitelný soubor C)**	**Klient (spustitelný soubor C)**
WebSphere MQ (koordinující XA-vyhovující správce databází) přístup k jedné databázi pomocí SQL	amqsxas0.sqc DB2 amqsxas0.ec Informix	amq0xas0.sqb	bez vzorku	bez vzorku
WebSphere MQ (koordinující XA-kompatibilní správce databází) přístup k dvěma databázím pomocí jazyka SQL	amqsxag0.c amqsxab0.sqc DB2 amqsxaf0.sqc DB2	amq0xag0.cbl amq0xab0.sqb amq0xaf0.sqb	bez vzorku	bez vzorku
Transakce TUXEDO pro vložení zpráv	amqstxpx	bez vzorku	bez vzorku	bez vzorku
Transakce TUXEDO pro získání zpráv	amqstxgx	bez vzorku	bez vzorku	bez vzorku
Server pro TUXEDO	amqstxsx	bez vzorku	bez vzorku	bez vzorku
obslužná rutina fronty nedoručených zpráv	Adresář `./tools/c/Samples/dlq` (“1” na stránce 104)	bez vzorku	amqsdlq	bez vzorku
Z klienta WebSphere MQ MQI, který vloží zprávu	bez vzorku	bez vzorku	bez vzorku	amqsputc
Z klienta WebSphere MQ MQI se získáním zprávy	bez vzorku	bez vzorku	bez vzorku	amqsgetc
Připojení ke správci front pomocí příkazu MQCONNX	amqscnxc	bez vzorku	bez vzorku	amqscnxc
Použití ukončení rozhraní API	amqsaxe0	bez vzorku	amqsaxe	bez vzorku
Vyrovnávání zátěže klastru	amqswlm0	bez vzorku	amqswlm	bez vzorku
Rutiny zabezpečení SSPI	amqsspin	bez vzorku	amqrspin.dll	amqrspin.dll
Asynchronně vkládání zpráv a získání stavu pomocí volání MQSTAT	amqsapt0	bez vzorku	amqsap	amqsaptc
Opakovaně připojitelní klienti	amqsphak amqsgak amqsmhak	bez vzorku	Nelze použít	amqsphak amqsgak amqsmhak
Použití spotřebitelů zpráv k asynchronnímu příjmu zpráv z více front	amqscbf0	bez vzorku	amqscbf	amqscbfc
Určení informací o připojení SSL/TLS pro MQCONNX	amqssslc	bez vzorku	nelze použít	amqssslc

Tabulka 16. Ukázkové programy produktu IBM WebSphere MQ for Windows demonstrují použití rozhraní MQI (C a COBOL) (pokračování)

Technika	C (zdroj)	COBOL (zdroj)	Server (spustitelný soubor C)	Klient (spustitelný soubor C)

Notes:

1. Zdroj pro popisovač fronty nedoručených zpráv se skládá z několika souborů a je uveden v samostatném adresáři.

Ukázky jazyka Visual Basic pro produkt IBM WebSphere MQ for Windows

Toto téma obsahuje techniky, které demonstrují ukázkové programy Visual Basic pro produkt IBM WebSphere MQ for Windows.

Tabulka 17 na stránce 104 zobrazuje techniky demonstrovány ukázkovými programy produktu IBM WebSphere MQ for Windows .

Projekt může obsahovat několik souborů. Když otevřete projekt v rámci Visual Basic, ostatní soubory se načtou automaticky. Nejsou k dispozici žádné spustitelné programy.

Všechny vzorové projekty, kromě mqtrivc.vbp, jsou nastaveny pro práci se serverem IBM WebSphere MQ . Informace o tom, jak lze změnit ukázkové projekty pro práci s klienty produktu IBM WebSphere MQ , naleznete v tématu “Příprava programů jazyka Visual Basic v systému Windows” na stránce 447.

Tabulka 17. IBM WebSphere MQ pro ukázkové programy systému Windows demonstrující použití rozhraní MQI (Visual Basic)

Technika	Název souboru projektu
Vložení zpráv pomocí volání MQPUT	amqsputb.vbp
Získávání zpráv pomocí volání MQGET	amqsgetb.vbp
Procházení fronty pomocí volání MQGET	amqsbcgb.vbp
Jednoduchá MQGET a ukázka MQPUT (klient)	mqtrivc.vbp
Jednoduchá MQGET a ukázka MQPUT (server)	mqtrivs.vbp
Vložení a získání řetězců a uživatelem definovaných struktur pomocí MQPUT a MQGET	strings.vbp
Použití struktur PCF ke spuštění a zastavení kanálu	pcfsamp.vbp
Vytvoření fronty s použitím rozhraní MQAI	amqsaicq.vbp
Zobrazení seznamu front správce front pomocí rozhraní MQAI	amqsailq.vbp
Monitorování událostí pomocí rozhraní MQAI	amqsaiem.vbp

Příprava a spuštění ukázkových programů

Konfigurujte svého 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 že byl spuštěn. Určete, zda jsou záznamy ověřování kanálu již povoleny, takto:

```
DISPLAY QMGR CHLAUTH
```

Tato úloha očekává, že jsou povoleny záznamy ověření kanálu. Pokud se jedná o správce front používaný ostatními uživateli a aplikacemi, změna tohoto nastavení bude mít vliv na všechny ostatní uživatele a aplikace. Pokud váš správce front nepoužívá záznamy ověření kanálu, může být krok “4” na stránce 105 nahrazen alternativní metodou ověření (například uživatelská procedura zabezpečení), která nastaví MCAUSER na *neprivilegované-id-uživatele* , které získáte v kroku “1” na stránce 105.

Musíte vědět, jaký název kanálu očekává vaše aplikace, aby mohla být aplikace povolena pro použití kanálu. Je třeba také vědět, které objekty, například fronty nebo témata, očekává, že aplikace bude používat aplikaci tak, aby je bylo možné používat.

Informace o této úloze

Tato úloha vytvoří neprivilegované ID uživatele, které má být použito pro aplikaci klienta, která se připojuje ke správci front. Přístup pro klientskou aplikaci je udělen pouze k tomu, aby mohl používat kanál, který potřebuje, a frontu, kterou potřebuje k použití tohoto ID uživatele.

Postup

1. Získejte ID uživatele na systému, na kterém je spuštěný správce front. Pro tuto úlohu nesmí být toto ID uživatele privilegovaný administrativní uživatel. Toto ID uživatele bude oprávněním, pod kterým bude spuštěno připojení klienta ve správci front.
2. Spusťte program listener s následujícími příkazy, kde:

   qmgr je název vašeho správce front
   nnnn je číslo vybraného portu

   a) Pro systémy UNIX a Windows :

   ```
   runmqlsr -t tcp -m qmgr -p nnnn
   ```

3. Používá-li aplikace SYSTEM.DEF.SVRCONN je tento kanál již definován. Pokud vaše aplikace používá jiný kanál, vytvořte jej zadáním příkazu MQSC:

   ```
   DEFINE CHANNEL('channel-name') CHLTYPE(SVRCONN) TRPTYPE(TCP) +
   DESCR('Channel for use by sample programs')
   ```

   název-kanálu je název vašeho kanálu.
4. Vytvořte pravidlo pro ověření kanálu, které umožňuje použití kanálu zadáním příkazu MQSC pouze zadáním adresy IP vašeho klienta:

   ```
   SET CHLAUTH('channel-name') TYPE(ADDRESSMAP) ADDRESS('client-machine-IP-address') +
   MCAUSER('non-privileged-user-id')
   ```

   název-kanálu je název vašeho kanálu.
   client-machine-IP-address je adresa IP vašeho klientského systému.
   Je-li ukázková aplikace klienta spuštěna na stejném počítači jako správce front, použijte adresu IP '127.0.0.1', pokud se vaše aplikace chystá připojit pomocí 'localhost'. Pokud se chystáte připojit několik různých klientských počítačů, můžete použít vzor nebo rozsah místo jedné IP adresy. Podrobnosti viz Generické adresy IP .
   non-privileged-user-id je ID uživatele, které jste získali v kroku “1” na stránce 105
5. Používá-li aplikace SYSTEM.DEFAULT.LOCAL.QUEUE je tato fronta již definována. Pokud vaše aplikace používá jinou frontu, vytvořte ji zadáním příkazu MQSC:

   ```
   DEFINE QLOCAL('queue-name') DESCR('Queue for use by sample programs')
   ```

   queue-name je název vaší fronty.
6. Udělte přístup pro připojení k správci front a dotazujte jej:

   a) Pro systémy UNIX aWindows zadejte příkazy MQSC:

```
SET AUTHREC OBJTYPE(QMGR) PRINCIPAL('non-privileged-user-id') +
AUTHADD(CONNECT, INQ)
```

non-privileged-user-id je ID uživatele, které jste získali v kroku “1” na stránce 105

7. Je-li vaše aplikace aplikací typu point-to-point, znamená to použití front, udělení přístupu k povolení a odesílání a získání zpráv pomocí vaší fronty pomocí ID uživatele, který má být použit, vydáním příkazů MQSC:

   a) Pro systémy UNIX a Windows zadejte příkazy MQSC:

```
SET AUTHREC PROFILE('queue-name') OBJTYPE(QUEUE) +
PRINCIPAL('non-privileged-user-id') AUTHADD(PUT, GET, INQ, BROWSE)
```

   queue-name je název vaší fronty.

   non-privileged-user-id je ID uživatele, které jste získali v kroku “1” na stránce 105

8. Je-li vaše aplikace aplikací typu publikování-odběr, která využívá témata, udělte přístup k povolení publikování a přihlášení k odběru tématu pomocí ID uživatele, který má být použit, zadáním příkazů MQSC:

   a) Pro systémy UNIX a Windows zadejte příkazy MQSC:

```
SET AUTHREC PROFILE('SYSTEM.BASE.TOPIC') OBJTYPE(TOPIC) +
PRINCIPAL('non-privileged-user-id') AUTHADD(PUB, SUB)
```

   non-privileged-user-id je ID uživatele, které jste získali v kroku “1” na stránce 105

   Tato akce poskytne uživateli *neprivilegované-ID-uživatele* přístup k libovolnému tématu ve stromu témat, případně můžete definovat objekt tématu pomocí produktu **DEFINE TOPIC** a udělit přístup pouze k části stromu témat odkazovaného daným objektem tématu. Podrobnosti najdete v tématu Řízení přístupu uživatelů k tématům .

Jak pokračovat dále

Klientská aplikace se nyní může připojit ke správci front a odesílat nebo přijímat zprávy s použitím fronty.

Související úlohy

Poskytnutí přístupu k objektu WebSphere MQ v systémech UNIX nebo Linux a v systému Windows

Související odkazy

SET CHLAUTH
Definovat kanál
DEFINOVAT QLOCAL
SET AUTHREC

Příprava a spuštění ukázkových programů v systémech UNIX

Tabulka 18. Kde najít ukázky pro produkt WebSphere MQ na systémech UNIX and Linux

Obsah	**Adresář**
Zdrojové soubory	*MQ_INSTALLATION_PATH*/samp
zdrojové soubory obslužné rutiny fronty nedoručených zpráv	*MQ_INSTALLATION_PATH*/samp/dlq
spustitelné soubory	*MQ_INSTALLATION_PATH*/samp/bin
Produkt *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.	

Ukázkové soubory produktu WebSphere MQ na systémech UNIX and Linux se nacházejí v adresářích uvedených v parametru Tabulka 18 na stránce 106 , pokud byly při instalaci použity výchozí hodnoty. Chcete-li ukázky spustit, buď použijte spustitelné verze dodané, nebo zkompilujte zdrojové verze

jako jakékoli jiné aplikace, a to pomocí kompilátoru ANSI. Informace o tom, jak to provést, najdete v tématu “Spuštění ukázkových programů” na stránce 107.

Příprava a spuštění ukázkových programů v systémech Windows

Tabulka 19. Kde najít ukázky pro produkt WebSphere MQ for Windows

Obsah	**Adresář**
Zdrojový kód C	*MQ_INSTALLATION_PATH*\Tools\C\Samples
Zdrojový kód pro ukázku obslužné rutiny dead-letter	*MQ_INSTALLATION_PATH*\Tools\C\Samples\DLQ
Zdrojový kód COBOL	*MQ_INSTALLATION_PATH*\Tools\Cobol \ Samples
Spustitelné soubory C [1]	*MQ_INSTALLATION_PATH*\ Tools\C\Samples \ Bin (32bitové verze) *MQ_INSTALLATION_PATH*\ Tools\C\Samples\Bin64 (64bitové verze)
Ukázkové soubory MQSC	*MQ_INSTALLATION_PATH*\Tools\MQSC\Samples
Zdrojový kód Visual Basic	*MQ_INSTALLATION_PATH*\Tools\VB\SampVB6
Ukázky .NET	*MQ_INSTALLATION_PATH*\Tools\dotnet \ Samples

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

Poznámka:

1. 64bitové verze jsou k dispozici pro některé ukázky spustitelných souborů jazyka C.

Ukázkové soubory produktu WebSphere MQ for Windows se nacházejí v adresářích uvedených v produktu Tabulka 19 na stránce 107 , pokud byly při instalaci použity výchozí hodnoty; instalační jednotka je standardně < c: >. Chcete-li spustit ukázky, buď použijte spustitelné verze dodané nebo zkompilujte zdrojové verze jako všechny ostatní aplikace produktu WebSphere MQ for Windows . Informace o tom, jak to provést, viz “Spuštění ukázkových programů” na stránce 107.

Spuštění ukázkových programů

Zvažte použití tohoto tématu při spouštění ukázkových programů na různých platformách.

Před spuštěním kteréhokoli z ukázkových programů vytvořte správce front a nastavte výchozí definice. To je vysvětleno v části Administrace.

Na platformách Windows, UNIX a Linux

Ukázky potřebují sadu front, se kterými se bude pracovat. Buď použijte vlastní fronty, nebo spusťte ukázkový soubor MQSC amqscos0.tst pro vytvoření sady.

Chcete-li to provést na systémech UNIX and Linux , zadejte:

- `runmqsc QManagerName <amqscos0.tst >/tmp/sampobj.out`

Zkontrolujte soubor `sampobj.out` , abyste se ujistili, že se nevyskytly žádné chyby.

Chcete-li to provést na systémech Windows , zadejte:

- `runmqsc QManagerName  <amqscos0.tst > sampobj.out`

Zkontrolujte soubor `sampobj.out` , abyste se ujistili, že se nevyskytly žádné chyby. Tento soubor se nachází ve vašem aktuálním adresáři.

Nyní můžete spustit ukázkové aplikace. Zadejte název ukázkové aplikace následovanou libovolněmi parametry, například:

- `amqsput myqueue qmanagername`

kde `myqueue` je název fronty, na kterou se mají zprávy vložit, a `qmanagername` je správce front, který vlastní `myqueue`.

Informace o parametrech, které každý z nich očekává, naleznete v popisu jednotlivých ukázek.

Délka názvu fronty

U ukázkových programů v jazyce COBOL při předávání názvů front jako parametrů je třeba zadat 48 znaků a v případě potřeby vyplňující prázdné znaky. Cokoli jiného než 48 znaků způsobí, že program selže s kódem příčiny 2085.

Příklady Inquire, Set a Echo

Pro příklady Inquire, Set a Echo spustí ukázkové definice verze C těchto ukázek.

Chcete-li, aby verze jazyka COBOL byly změněny, musíte změnit definice procesu:

- SYSTEM.SAMPLE.INQPROCESS
- SYSTEM.SAMPLE.SETPROCESS
- SYSTEM.SAMPLE.ECHOPROCESS

Na systémech Windowsse systémy UNIX and Linux takto upravují úpravou souboru `amqscos0.tst` a změnou názvů spustitelných souborů jazyka C na názvy spustitelných souborů v jazyce COBOL před použitím příkazu **runmqsc** , jak bylo ukázáno výše.

Ukázkový program uživatelské procedury rozhraní API

Ukázková uživatelská procedura rozhraní API vygeneruje trasování MQI do uživatelem určeného souboru s předponou definovanou v proměnné prostředí MQAPI_TRACE_LOGFILE.

Další informace o uživatelských procedurách rozhraní API najdete v tématu “Zápis a kompilace uživatelských procedur rozhraní API” na stránce 372.

Zdroj
amqsaxe0.c

Binární
amqsaxe

Konfigurace pro uživatelskou proceduru pro ukázku

1. Přidejte následující do souboru qm.ini .

   Platformy jiné než Windows

   ```
   ApiExitLocal:
      Sequence=100
      Function=EntryPoint
      Module=MQ_INSTALLATION_PATH/samp/bin/amqsaxe
      Name=SampleApiExit
   ```

   kde *MQ_INSTALLATION_PATH* představuje adresář, kde je nainstalován produkt IBM WebSphere 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 WebSphere MQ .

2. Nastavit proměnnou prostředí

```
MQAPI_TRACE_LOGFILE=/tmp/MqiTrace
```

3. Spusťte aplikaci.

   Výstupní soubory jsou vytvářeny v adresáři /tmp s názvy jako: `MqiTrace.<pid>.<tid>.log`

Ukázkový program asynchronní spotřeby

Ukázkový program amqscbf demonstruje použití příkazů MQCB a MQCTL ke spotřebování zpráv z více front asynchronně.

amqscbf je k dispozici jako zdrojový kód jazyka C a binární klient a server na platformách Windows, UNIX and Linux .

Program je spuštěn z příkazového řádku a používá následující volitelné parametry:

```
Usage: [Options] <Queue Name> { <Queue Name> }
  where Options are:
    -m <Queue Manager Name>
    -o <Open options>
    -r <Reconnect Type>
       d Reconnect Disabled
       r Reconnect
       m Reconnect Queue Manager
```

Chcete-li číst zprávy z více front, zadejte více než jeden název fronty (maximálně deset front je podporováno vzorkem).

Poznámka: *Typ opětovného připojení* je platný pouze pro klientské programy.

Příklad

Příklad zobrazuje příkaz amqscbf spuštěný jako serverový program, který čte jednu zprávu z produktu QL1 a poté je zastavován.

K vložení testovací zprávy do produktu QL1použijte Průzkumníka produktu WebSphere 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 parametr amqscbf demonstruje

Ukázka ukazuje, jak číst zprávy z více front v pořadí jejich příchodu. To bude vyžadovat mnohem více kódu pomocí synchronního příkazu MQGET. V případě asynchronní spotřeby není požadován žádný systém výzev a správa podprocesů a úložišť je prováděna produktem WebSphere MQ. Příklad "reálného světa" by se musel vypořádat s chybami; v ukázkových chybách jsou na konzolu odepsány chyby.

Vzorový kód má následující kroky:

1. Definujte jednotlivou funkci zpětného volání spotřeby zpráv,

```
void MessageConsumer(MQHCONN      hConn,
                     MQMD    * pMsgDesc,
                     MQGMO   * pGetMsgOpts,
                     MQBYTE  * Buffer,
```

```
                           MQCBC   * pContext)
{ ... }
```

2. Připojte se ke správci front,

```
MQCONNX(QMName,&cno,&Hcon,&CompCode,&CReason);
```

3. Otevřete vstupní fronty a přidružte je ke každé funkci zpětného volání `MessageConsumer` .

```
 MQOPEN(Hcon,&od,O_options,&Hobj,&OpenCode,&Reason);
 cbd.CallbackFunction = MessageConsumer;
 MQCB(Hcon,MQOP_REGISTER,&cbd,Hobj,&md,&gmo,&CompCode,&Reason);
```

`cbd.CallbackFunction` není třeba nastavovat pro každou frontu; jedná se pouze o vstupní pole. Ke každé frontě byste však mohli přidružit jinou funkci zpětného volání.

4. Spustit spotřebu zpráv,

```
MQCTL(Hcon,MQOP_START,&ctlo,&CompCode,&Reason);
```

5. Počkejte, až uživatel stiskne stisknutou klávesu Enter a poté ukončí spotřebu zpráv,

```
MQCTL(Hcon,MQOP_STOP,&ctlo,&CompCode,&Reason);
```

6. Nakonec se odpojte od správce front,

```
MQDISC(&Hcon,&CompCode,&Reason);
```

Ukázkový program Asynchronous Put

Zde se dozvíte o spuštění ukázky amqsapt a návrhu ukázkového programu Asynchronous Put.

Ukázkový program pro asynchronní vložení vkládá zprávy do fronty s použitím asynchronního volání MQPUT a poté načte informace o stavu pomocí volání MQSTAT. Název tohoto programu na různých platformách viz “Vlastnosti demonstrovány v ukázkových programech” na stránce 94 .

Spuštění ukázky amqsapt

Tento program může mít až 6 parametrů:

1. Název cílové fronty (povinné)
2. Název správce front (volitelný)
3. Volby otevření (volitelné)
4. Volby zavření (volitelné)
5. Název cílového správce front (volitelné)
6. Název dynamické fronty (volitelné)

Není-li určen správce front, připojí se parametr amqSapt k výchozímu správci front.

Návrh ukázkového programu Asynchronous Put

Program používá volání MQOPEN s dodanými volbami pro výstup, nebo s volbami MQOO_OUTPUT a MQOO_FAIL_IF_QUIESCING k otevření cílové fronty pro vkládání zpráv.

Pokud nemůže frontu otevřít, výstupem programu je chybová zpráva obsahující kód příčiny vrácený voláním MQOPEN. Chcete-li program ponechat jednoduchý, a to i při následných voláních MQI, program použije výchozí hodnoty pro celou řadu voleb.

Pro každý řádek vstupu program přečte text do vyrovnávací paměti a použije volání MQPUT s MQPMO_ASYNC_RESPONSE k vytvoření zprávy datagramu obsahující text této linky a asynchronně jej umístí do cílové fronty. Program pokračuje, dokud nedosáhne konce vstupu, nebo volání MQPUT selže. Pokud se program dostane na konec vstupu, zavře frontu pomocí volání MQCLOSE.

Program pak vydá volání MQSTAT, vrátí strukturu MQSTS a zobrazí zprávy obsahující počet úspěšně vložených zpráv, počet zpráv vložených s varováním a počet selhání.

Ukázkové programy Procházet

Při procházení ukázkových programů procházejte zprávy ve frontě pomocí volání MQGET.

Názvy těchto programů viz “Vlastnosti demonstrovány v ukázkových programech” na stránce 94 .

Návrh ukázkového programu pro procházení

Program otevře cílovou frontu pomocí volání MQOPEN s volbou MQOO_BROWSE. Pokud nemůže frontu otevřít, výstupem programu je chybová zpráva obsahující kód příčiny vrácený voláním MQOPEN.

Pro každou zprávu ve frontě používá program volání MQGET ke zkopírování zprávy z fronty a poté zobrazí data obsažená ve zprávě. Volání MQGET používá tyto volby:

PŘÍŠTĚ MQGMO_BROWSE_NEXT
Po volání MQOPEN je kurzor procházení umístěn logicky před první zprávou ve frontě, takže tato volba způsobí vrácení zprávy ***první*** při prvním vyvolání volání.

MQGMO_NO_WAIT
Program nečeká, pokud ve frontě nejsou žádné zprávy.

SOUBOR MQGMO_ACCEPT_TRUNCATED_MSG
Volání MQGET určuje vyrovnávací paměť pevné velikosti. Je-li zpráva delší než tato vyrovnávací paměť, program zobrazí oříznutou zprávu spolu s varováním, že zpráva byla zkrácena.

Tento program demonstruje, jak musíte vymazat pole *MsgId* a *CorrelId* struktury MQMD po každém volání MQGET, protože volání nastavuje tato pole na hodnoty obsažené ve zprávě, kterou načítá. Vymazání těchto polí znamená, že po sobě jdoucí příkazy MQGET načítají zprávy v pořadí, ve kterém jsou zprávy zadrženy ve frontě.

Program pokračuje na konci fronty; volání MQGET vrátí kód příčiny MQRC_NO_MSG_AVAILABLE a program zobrazí varovnou zprávu. Pokud se volání MQGET nezdaří, zobrazí program chybovou zprávu, která obsahuje kód příčiny.

Program poté zavře frontu pomocí volání MQCLOSE.

Systémy UNIX, Linux a Windows

Zvažte použití tohoto tématu, když se seznámíte se vzorovými programy Procházet na systémech UNIX, Linux a Windows .

Verze C programu má 2 parametry

1. Název zdrojové fronty (nezbytné)
2. Název správce front (volitelný)

Není-li správce front zadán, připojí se k výchozímu správci front. Zadejte například jednu z následujících možností:

- `amqsgbr myqueue qmanagername`
- `amqsgbrc myqueue qmanagername`
- `amq0gbr0 myqueue`

kde `myqueue` je název fronty, ze které se budou zobrazovat zprávy, a `qmanagername` je správce front, který vlastní `myqueue`.

Pokud při spuštění ukázky jazyka C vynecháte `qmanagername`, předpokládá se, že výchozí správce front vlastní tuto frontu.

Verze COBOL nemá žádné parametry. Připojuje se k výchozímu správci front, a když jej spustíte, budete vyzváni:

```
Please enter the name of the target queue
```

Zobrazí se pouze prvních 50 znaků každé zprávy, za nímž bude následovat `- - - truncated` , pokud se jedná o tento případ.

Ukázkový program prohlížeče

Ukázkový program prohlížeče čte a zapisuje jak deskriptor zprávy, tak pole obsahu zprávy ve všech zprávách ve frontě.

Ukázkový program je napsán jako utilita, ne jen aby demonstroval techniku. Názvy těchto programů viz “Vlastnosti demonstrovány v ukázkových programech” na stránce 94 .

Tento program vezme tyto parametry:

1. Název zdrojové fronty
2. Název správce front
3. Volitelný parametr pro vlastnosti.

První dva vstupní parametry pro tento program jsou povinné. Program můžete spustit například jedním z následujících způsobů:

- `amqsbcg myqueue qmanagername`
- `amqsbcgc myqueue qmanagername`

kde `myqueue` je název fronty, v níž mají být zprávy procházeny, a `qmanagername` je správce front, který vlastní `myqueue`.

Přečte každou zprávu z fronty a zapíše na standardní výstup následující výstup:

- Pole deskriptoru formátované zprávy
- Data zprávy (vypsána v hexadecimálním formátu a, je-li to možné, znakový formát)

Přípustné hodnoty pro parametr vlastnosti jsou:

Hodnota	Chování
0	Výchozí chování, protože to bylo pro V6. Vlastnosti doručené do aplikace závisí na atributu fronty *PropertyControl* , ze kterého se zpráva načítá.

1	Popisovač zprávy je vytvořen a použit s parametrem MQGET. Vlastnosti zprávy s výjimkou těch, které jsou obsaženy v deskriptoru (či rozšíření) zprávy, jsou zobrazeny podobným způsobem jako deskriptor zprávy. Příklad:
	`****Message properties****`
	`<property name> : <property value>`
	Nebo nejsou-li k dispozici žádné vlastnosti:
	`****Message properties****`
	`None`
	Číselné hodnoty jsou zobrazeny pomocí `printf`, hodnoty řetězce jsou uzavřeny v jednoduchých uvozovkách a bajtové řetězce jsou obklopeny X a jednoduchými uvozovkami, jako pro deskriptor zprávy.

2	Hodnota MQGMO_NO_PROPERTIES je určena tak, aby byly vráceny pouze vlastnosti deskriptoru zpráv.
3	Je zadán parametr MQGMO_PROPERTIES_FORCE_MQRFH2 , takže všechny vlastnosti jsou vráceny v datech zprávy.

Hodnota	Chování
4	Funkce MQGMO_PROPERTIES_COMPATIBILITY je určena tak, aby všechny vlastnosti mohly být vraceny v závislosti na tom, zda je zahrnuta vlastnost verze 6, jinak jsou vlastnosti zrušeny.

Program je omezen na tisk prvních 65535 znaků zprávy a selže s příčinou *zkrácená zpráva* , pokud je přečtena delší zpráva.

Příklad výstupu z tohoto obslužného programu najdete v příručce Administrace .

Ukázka transakce CICS

Ukázkový transakční program CICS je poskytován s názvem amqscic0.ccs pro zdrojový kód a amqscic0 pro spustitelnou verzi. Transakce můžete sestavovat pomocí standardních zařízení CICS .

Podrobnosti o příkazech potřebných pro vaši platformu naleznete v příručce “Sestavení aplikace IBM WebSphere MQ” na stránce 412 .

Transakce čte zprávy z přenosové fronty SYSTEM.SAMPLE.CICS.WORKQUEUE na výchozím správci front a umisťuje je do lokální fronty, jejíž název je obsažen v záhlaví přenosu zprávy. Veškerá selhání se odesílají do fronty SYSTEM.SAMPLE.CICS.DLQ.

Poznámka: K vytvoření těchto front a ukázkových vstupních front můžete použít vzorový skript MQSC amqscic0.tst .

Ukázkový program Connect

Ukázkový program Connect vám umožňuje prozkoumat volání MQCONNX a jeho volby od klienta. Ukázka se připojí ke správci front pomocí volání MQCONNX, inquires o názvu správce front s použitím volání MQINQ a zobrazí jej. Také se seznámíte se spuštěním ukázky amqscnxc.

Poznámka: Ukázkový program Connect je ukázkový klient. Můžete ji zkompilovat a spustit na serveru, ale funkce je smysluplná pouze na straně klienta a jsou dodány pouze spustitelné soubory klienta.

Spuštění ukázky amqscnxc

Syntaxe příkazového řádku ukázkového programu Connect je následující:

```
amqscnxc [-x ConnName [-c SvrconnChannelName]] [QMgrName]
```

Parametry jsou volitelné a jejich pořadí není důležité, kromě parametru QMgrName, který, je-li zadán, musí být poslední. Parametry jsou:

ConnName
Název připojení TCP/IP správce front serveru

Název SvrconnChannel
Název kanálu připojení serveru

QMgrName
Název cílového správce front

Nezadáte-li název připojení TCP/IP, bude MQCONNX zadán spolu s parametrem *ClientConnPtr* nastaveným na hodnotu NULL. Uvedete-li jméno připojení TCP/IP, ale ne kanál připojení k serveru (zpětné nastavení není povoleno), ukázka použije název SYSTEM.DEF.SVRCONN. Pokud neurčíte cílového správce front, bude se ukázka připojovat k kterémkoliv správci front, který naslouchá danému názvu připojení protokolu TCP/IP.

Poznámka: Zadáte-li otazník jako jediný parametr nebo zadáte-li nesprávné parametry, zobrazí se zpráva s vysvětlením, jak tento program používat.

Spustíte-li ukázku bez voleb příkazového řádku, bude obsah proměnné prostředí MQSERVER použit k určení informací o připojení. (V tomto příkladu je MQSERVER nastaven na `SYSTEM.DEF.SVRCONN/TCP/machine.site.company.com`.) Zobrazí se výstup podobný tomuto:

```
Sample AMQSCNXC start
Connecting to the default queue manager
with no client connection information specified.
Connection established to queue manager machine

Sample AMQSCNXC end
```

Pokud spustíte ukázku a zadáte-li název připojení TCP/IP a název kanálu připojení serveru, ale žádný název cílového správce front, jako je tento:

```
amqscnxc -x machine.site.company.com -c SYSTEM.ADMIN.SVRCONN
```

je použit výchozí název správce front a vidíte výstup podobný tomuto:

```
Sample AMQSCNXC start
Connecting to the default queue manager
using the server connection channel SYSTEM.ADMIN.SVRCONN
on connection name machine.site.company.com.
Connection established to queue manager MACHINE

Sample AMQSCNXC end
```

Pokud spustíte ukázku a zadáte-li název připojení TCP/IP a název cílového správce front, jako je tento:

```
amqscnxc -x machine.site.company.com MACHINE
```

zobrazí se výstup podobný tomuto:

```
Sample AMQSCNXC start
Connecting to queue manager MACHINE
using the server connection channel SYSTEM.DEF.SVRCONN
on connection name machine.site.company.com.
Connection established to queue manager MACHINE

Sample AMQSCNXC end
```

Ukázkový program pro převod dat

Ukázkový program pro převod dat je kostra uživatelské rutiny pro převod dat. Informace o návrhu ukázky pro převod dat.

Názvy těchto programů viz “Vlastnosti demonstrovány v ukázkových programech” na stránce 94 .

Návrh souboru pro převod dat

Každá uživatelská procedura pro převod dat převádí jeden pojmenovaný formát zpráv. Tato kostra je určena jako obálka pro fragmenty kódu generované obslužným programem pro generování dat ukončení převodu dat.

Obslužný program vytváří jeden fragment kódu pro každou strukturu dat; několik takových struktur tvoří formát, takže se do této kostry přidá několik fragmentů kódu, aby bylo možné vytvořit rutinu tak, aby bylo možné provádět převod dat celého formátu.

Program potom zkontroluje, zda je konverze úspěšná nebo neúspěšná, a vrátí hodnoty vyžadované volajícímu.

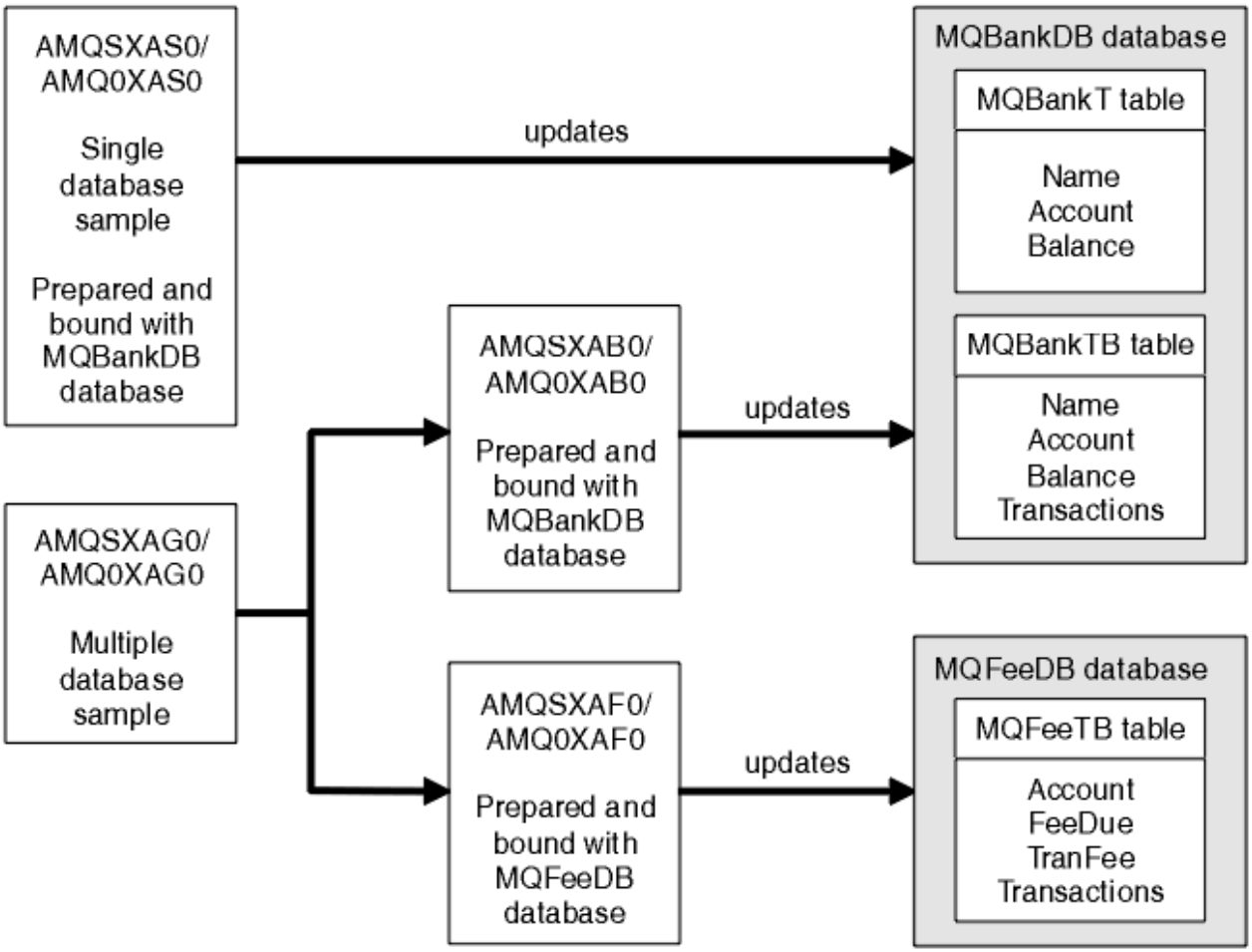
Ukázky koordinace databází

K dispozici jsou dva ukázky, které ukazují, jak produkt WebSphere MQ může koordinovat aktualizace WebSphere MQ a aktualizace databáze v rámci stejné pracovní jednotky.

Tyto vzorky jsou:

1. AMQSXAS0 (v jazyce C) nebo AMQ0XAS0 (v COBOLu), které aktualizuje jednu databázi v rámci pracovní jednotky WebSphere MQ .
2. AMQSXAG0 (v jazyce C) nebo AMQ0XAG0 (v COBOLu), AMQSXAB0 (v jazyce C) nebo AMQ0XAB0 (v COBOLu) a AMQSXAF0 (v jazyce C) nebo AMQ0XAF0 (v COBOLu), které dohromady aktualizují dvě databáze v rámci pracovní jednotky WebSphere MQ , které ukazují, jak lze přistupovat k více databázím. Tyto ukázky jsou k dispozici pro zobrazení použití volání MQBEGIN, smíšených volání SQL a WebSphere MQ a kde a kdy a kdy se má připojit k databázi.

Produkt Obrázek 18 na stránce 115 zobrazuje, jak jsou použité ukázky použity k aktualizaci databází:



Obrázek 18. Ukázky koordinace databází

Programy přečtou zprávu z fronty (pod synchronizačním bodem) a pak pomocí informací ve zprávě získají příslušné informace z databáze a aktualizují ji. Pak se vytiskne nový stav databáze.

Logika programu je následující:

1. Použijte název vstupní fronty z argumentu programu
2. Připojit se k výchozímu správci front (nebo volitelně k zadanému názvu v jazyce C) pomocí MQCONN
3. Otevření fronty (použití funkce MQOPEN) pro vstup při neúspěších
4. Spuštění jednotky práce pomocí MQBEGIN

5. Získat další zprávu (pomocí příkazu MQGET) z fronty pod synchronizačním bodem
6. Získat informace z databází
7. Aktualizovat informace z databází
8. Potvrdit změny pomocí MQCMIT
9. Vytisknout aktualizované informace (žádná zpráva, která má být k dispozici, se počítá jako selhání, a konec smyčky)
10. Zavřít frontu pomocí příkazu MQCLOSE
11. Odpojit od fronty pomocí MQDISC

Kurzory SQL se používají ve vzorcích, takže čtení z databází (tj. více instancí) je uzamčeno během zpracování zprávy, což umožňuje simultánní spuštění více instancí těchto programů. Kurzory jsou explicitně otevřeny, ale volání MQCMIT je implicitně zavřeny.

Jedna ukázka databáze (AMQSXAS0 nebo AMQ0XAS0) nemá žádné příkazy SQL CONNECT a připojení k databázi je implicitně provedeno produktem WebSphere MQ s voláním MQBEGIN. Ukázka více databází (AMQSXAG0 nebo AMQ0XAG0, AMQSXAB0 nebo AMQ0XAB0a AMQSXAF0 nebo AMQ0XAF0) obsahuje příkazy SQL CONNECT, protože některé databázové produkty povolují pouze jedno aktivní připojení. Pokud se nejedná o tento případ pro daný databázový produkt nebo pokud přistupujete k jedné databázi ve více databázových produktech, lze příkazy SQL CONNECT odstranit.

Ukázky jsou připraveny s databázovým produktem IBM DB2 , takže je možná budete muset upravit tak, aby fungovaly s dalšími databázovými produkty.

Kontrola chyb SQL používá rutiny v UTIL.C a CHECKERR.CBL poskytla produkt DB2. Musí být kompilovány nebo nahrazeny před kompilací a propojením.

Poznámka: Pokud používáte zdroj jazyka COBOL Micro Focus, CHECKERR.MFC pro kontrolu chyb SQL, musíte změnit ID programu na velká písmena, což je CHECKERR, aby AMQ0XAS0 propojí správně.

Vytvoření databází a tabulek

Vytvořte databáze a tabulky před kompilací ukázek.

Chcete-li vytvořit databáze, použijte obvyklou metodu pro váš databázový produkt, například:

```
  DB2 CREATE DB MQBankDB
  DB2 CREATE DB MQFeeDB
```

Vytvořte tabulky pomocí příkazů SQL následujícím způsobem:

V C:

```
EXEC SQL CREATE TABLE MQBankT(Name         VARCHAR(40) NOT NULL,
                              Account      INTEGER     NOT NULL,
                              Balance      INTEGER     NOT NULL,
                              PRIMARY KEY (Account));

EXEC SQL CREATE TABLE MQBankTB(Name         VARCHAR(40) NOT NULL,
                               Account      INTEGER     NOT NULL,
                               Balance      INTEGER     NOT NULL,
                               Transactions INTEGER,
                               PRIMARY KEY (Account));

EXEC SQL CREATE TABLE MQFeeTB(Account      INTEGER     NOT NULL,
                              FeeDue       INTEGER     NOT NULL,
                              TranFee      INTEGER     NOT NULL,
                              Transactions INTEGER,
                              PRIMARY KEY (Account));
```

V jazyce COBOL:

```
 EXEC SQL CREATE TABLE
   MQBankT(Name      VARCHAR(40) NOT NULL,
           Account   INTEGER     NOT NULL,
           Balance   INTEGER     NOT NULL,
```

```
           PRIMARY KEY (Account))
   END-EXEC.

EXEC SQL CREATE TABLE
  MQBankTB(Name      VARCHAR(40) NOT NULL,
           Account   INTEGER     NOT NULL,
           Balance   INTEGER     NOT NULL,
           Transactions INTEGER,
           PRIMARY KEY (Account))
  END-EXEC.

EXEC SQL CREATE TABLE
  MQFeeTB(Account      INTEGER     NOT NULL,
          FeeDue       INTEGER     NOT NULL,
          TranFee      INTEGER     NOT NULL,
          Transactions INTEGER,
          PRIMARY KEY (Account))
  END-EXEC.
```

Zadejte data do tabulek pomocí příkazů SQL následujícím způsobem:

```
EXEC SQL INSERT INTO MQBankT VALUES ('Mr Fred Bloggs',1,0);
EXEC SQL INSERT INTO MQBankT VALUES ('Mrs S Smith',2,0);
EXEC SQL INSERT INTO MQBankT VALUES ('Ms Mary Brown',3,0);
.
EXEC SQL INSERT INTO MQBankTB VALUES ('Mr Fred Bloggs',1,0,0);
EXEC SQL INSERT INTO MQBankTB VALUES ('Mrs S Smith',2,0,0);
EXEC SQL INSERT INTO MQBankTB VALUES ('Ms Mary Brown',3,0,0);
.
EXEC SQL INSERT INTO MQFeeTB VALUES (1,0,50,0);
EXEC SQL INSERT INTO MQFeeTB VALUES (2,0,50,0);
EXEC SQL INSERT INTO MQFeeTB VALUES (3,0,50,0);
.
```

Poznámka: Pro COBOL použijte stejné příkazy SQL, ale přidejte END_EXEC na konec každého řádku.

Předkompilace, kompilace a propojování ukázek

Seznamte se s předkompilací, kompilací a propojením ukázek v jazycích C a COBOL.

Předkompilujte soubory .SQC (v jazyce C) a soubory .SQB (v jazyce COBOL) a svážete je s příslušnou databází pro vytvoření souborů .C nebo .CBL. Chcete-li to provést, použijte typickou metodu pro váš databázový produkt.

Předkompilace v jazyce C

```
db2 connect to MQBankDB
db2 prep AMQSXAS0.SQC
db2 connect reset

db2 connect to MQBankDB
db2 prep AMQSXAB0.SQC
db2 connect reset

db2 connect to MQFeeDB
db2 prep AMQSXAF0.SQC
db2 connect reset
```

Předkompilace v jazyce COBOL

```
db2 connect to MQBankDB
db2 prep AMQ0XAS0.SQB bindfile target ibmcob
db2 bind AMQ0XAS0.BND
db2 connect reset

db2 connect to MQBankDB
db2 prep AMQ0XAB0.SQB bindfile target ibmcob
db2 bind AMQ0XAB0.BND
db2 connect reset

db2 connect to MQFeeDB
```

```
db2 prep AMQ0XAF0.SQB bindfile target ibmcob
db2 bind AMQ0XAF0.BND
db2 connect reset
```

Kompilace a propojení

Následující ukázkové příkazy používají symboly *<DB2TOP>* a *MQ_INSTALLATION_PATH*. *<DB2TOP>* představují instalační adresář pro produkt DB2 . *MQ_INSTALLATION_PATH* Představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

- V systému AIXje cesta k adresáři následující:

```
/usr/lpp/db2_05_00
```

- V systému HP-UX a Solaris je cesta k adresáři:

```
/opt/IBMdb2/V5.0
```

- Na systémech Windows závisí cesta k adresáři na cestě zvolené při instalaci produktu. Pokud jste zvolili výchozí nastavení, je cesta následující:

```
c:\sqllib
```

Poznámka: Před zadáním příkazu propojení na systémy Windows se ujistěte, že proměnná prostředí LIB obsahuje cesty k knihovnám DB2 a WebSphere MQ .

Zkopírujte následující soubory do dočasného adresáře:

- Soubor `amqsxag0.c` z instalace produktu WebSphere MQ

  Poznámka: Tento soubor lze nalézt v následujících adresářích:

  – Na systémech UNIX and Linux :

  ```
  MQ_INSTALLATION_PATH/samp/xatm
  ```

  – Na systémech Windows :

  ```
  MQ_INSTALLATION_PATH\tools\c\samples\xatm
  ```

- Soubory `.c` , které jste získali předkompilací zdrojových souborů `.sqc` , `amqsxas0.sqc` , `amqsxaf0.sqc` a `amqsxab0.sqc`
- Soubory `util.c` a `util.h` z vaší instalace DB2 .

  Poznámka: Tyto soubory lze nalézt v adresáři:

  ```
  <DB2TOP>/samples/c
  ```

Sestavte objektové soubory pro každý soubor `.c` pomocí následujícího příkazu kompilátoru pro platformu, kterou používáte:

- AIX

```
xlc_r -IMQ_INSTALLATION_PATH/inc -I
<DB2TOP>/include -c -o
 <FILENAME>.o <FILENAME>.c
```

- HP-UX

```
cc -Aa +z -IMQ_INSTALLATION_PATH/inc -I
```

```
<DB2TOP>/include -c -o
 <FILENAME>.o <FILENAME>.c
```

- Solaris

```
cc -Aa -KPIC -mt -IMQ_INSTALLATION_PATH
/inc -I<DB2TOP>/include -c -o
 <FILENAME>.o <FILENAME>.c
```

- systémy Windows

```
cl /c /IMQ_INSTALLATION_PATH\tools\c\include /I
<DB2TOP>\include
 <FILENAME>.c
```

Sestavte spustitelný soubor produktu amqsxag0 pomocí následujícího příkazu odkazu pro platformu, kterou používáte:

- AIX

```
xlc_r -H512 -T512 -L<DB2TOP>/lib -ldb2 -LMQ_INSTALLATION_PATH/lib
 -lmqm util.o amqsxaf0.o amqsxab0.o amqsxag0.o -o amqsxag0
```

- HP-UX , revize 11i

```
ld -E -L<DB2TOP>/lib -ldb2 -LMQ_INSTALLATION_PATH/lib -lmqm -lc -lpthread -lcl
 /lib/crt0.o util.o amqsxaf0.o amqsxab0.o amqsxag0.o -o amqsxag0
```

- Solaris

```
cc -mt -L<DB2TOP>/lib -ldb2 -LMQ_INSTALLATION_PATH/lib
 -lmqm -lthread -lsocket -lc -lnsl -ldl util.o
  amqsxaf0.o amqsxab0.o amqsxag0.o -o amqsxag0
```

- systémy Windows

```
link util.obj amqsxaf0.obj amqsxab0.obj amqsxag0.obj mqm.lib db2api.lib
 /out:amqsxag0.exe
```

Sestavte spustitelný soubor produktu amqsxas0 s použitím následujících příkazů pro kompilaci a propojení pro platformu, kterou používáte:

- AIX

```
xlc_r -H512 -T512 -L<DB2TOP>/lib -ldb2
 -LMQ_INSTALLATION_PATH/lib -lmqm util.o amqsxas0.o -o amqsxas0
```

- HP-UX , revize 11i

```
ld -E -L<DB2TOP>/lib -ldb2 -LMQ_INSTALLATION_PATH/lib -lmqm -lc -lpthread
 -lcl /lib/crt0.o util.o amqsxas0.o -o amqsxas0
```

- Solaris

```
cc -mt -L<DB2TOP>/lib -ldb2-LMQ_INSTALLATION_PATH/lib
 -lqm -lthread -lsocket -lc -lnsl -ldl util.o
  amqsxas0.o -o amqsxas0
```

- systémy Windows

```
link util.obj amqsxas0.obj mqm.lib db2api.lib /out:amqsxas0.exe
```

Další informace

Pracujete-li v systému AIX nebo HP-UX a chcete přistupovat k databázi Oracle, použijte kompilátor xlc_r a odkaz na soubor libmqm_r.a.

Spuštění ukázek

V této části se dozvíte, jak nakonfigurovat správce front před spuštěním ukázek koordinace databází v jazycích C a COBOL.

Před spuštěním ukázek nakonfigurujte správce front s použitím databázového produktu, který používáte. Další informace o tom, jak to provést, viz "Scénář 1: Správce front provádí koordinaci" na stránce 41.

Následující titulky poskytují informace o tom, jak spouštět ukázky v jazycích C a COBOL:

- "Ukázky jazyka C" na stránce 120
- "Ukázky jazyka COBOL" na stránce 121

Ukázky jazyka C

Zprávy musí být v následujícím formátu, aby mohly být čteny z fronty:

```
UPDATE Balance change=nnn WHERE Account=nnn
```

AMQSPUT lze použít k umístění zpráv do fronty.

Ukázky koordinace databází mají dva parametry:

1. Název fronty (povinné)
2. Název správce front (volitelný)

Za předpokladu, že jste vytvořili a nakonfigurovali správce front pro jednu ukázku databáze s názvem singDBQM, s frontou s názvem singDBQ, zvyšte hodnotu účtu pana Fred Bloggs o 50 takto:

```
AMQSPUT singDBQ singDBQM
```

Pak klíč v této zprávě:

```
UPDATE Balance change=50 WHERE Account=1
```

Do fronty můžete vložit více zpráv.

```
AMQSXAS0 singDBQ singDBQM
```

Pak se vytiskne aktualizovaný stav účtu pana Freda Bloggse.

Za předpokladu, že jste vytvořili a nakonfigurovali správce front pro ukázku více databází s názvem multDBQM, s frontou nazvanou multDBQ, snižujete účet paní Mary Brownové o 75 takto:

```
AMQSPUT multDBQ multDBQM
```

Pak klíč v této zprávě:

```
UPDATE Balance change=-75 WHERE Account=3
```

Do fronty můžete vložit více zpráv.

```
AMQSXAG0 multDBQ multDBQM
```

Pak se vytiskne aktualizovaný stav účtu paní Mary Brownové.

Ukázky jazyka COBOL

Zprávy musí být v následujícím formátu, aby mohly být čteny z fronty:

```
UPDATE Balance change=snnnnnnnn WHERE Account=nnnnnnnn
```

Pro jednoduchost musí `Balance change` být číslo se znakem osmdesát a `Account` musí být osminásobné číslo.

Ukázku AMQSPUT lze použít k umístění zpráv do fronty.

Ukázky nepřijímají žádné parametry a používají výchozího správce front. Lze jej nakonfigurovat tak, aby v libovolném okamžiku mohl být spuštěn pouze jeden z ukázek. Za předpokladu, že jste nakonfigurovali výchozího správce front pro jednu ukázku databáze s frontou s názvem singDBQ, zvyšte hodnotu účtu pana Fred Bloggs o 50 takto:

```
AMQSPUT singDBQ
```

Pak klíč v této zprávě:

```
UPDATE Balance change=+00000050 WHERE Account=00000001
```

Do fronty můžete vložit více zpráv:

```
AMQ0XAS0
```

Zadejte název fronty:

```
singDBQ
```

Pak se vytiskne aktualizovaný stav účtu pana Freda Bloggse.

Za předpokladu, že jste nakonfigurovali výchozího správce front pro více databázových ukázek s frontou s názvem multDBQ, jste odhadli účet paní Mary Mary o 75 takto:

```
AMQSPUT multDBQ
```

Pak klíč v této zprávě:

```
UPDATE Balance change=-00000075 WHERE Account=00000003
```

Do fronty můžete vložit více zpráv:

```
AMQ0XAG0
```

Zadejte název fronty:

```
multDBQ
```

Pak se vytiskne aktualizovaný stav účtu paní Mary Brownové.

Ukázka obslužné rutiny fronty nedoručených zpráv

Je k dispozici ukázkový popisovač fronty nedoručených zpráv, název spustitelné verze je amqsdlq. Chcete-li používat obslužnou rutinu fronty nedoručených zpráv, která se liší od proměnné RUNMQDLQ, je zdroj ukázky k dispozici pro použití jako základ.

Ukázka je podobná obslužnému programu na dead-letter v produktu, ale trasování a hlášení chyb se liší. K dispozici jsou dvě proměnné prostředí:

TRASOVÁNÍ DOQ_TRACE
Nastavit na YES nebo ano, chcete-li zapnout trasování

ODQ_ZPR
Nastavte na název souboru obsahujícího chybové a informační zprávy. Poskytnutý soubor se nazývá amqsdlq.msg.

Tyto proměnné je třeba v závislosti na platformě zobrazit pomocí příkazů **export** nebo **set** , a to v závislosti na platformě; trasování je vypnuto pomocí příkazu **unset** .

Soubor s chybovou zprávou amqsdlq.msgmůžete upravit tak, aby vyhovoval vašim požadavkům. Ukázka umísťuje zprávy do souboru stdout, ***not*** do souboru protokolu chyb produktu WebSphere MQ .

Příručka Administrace nebo *System Management Guide* pro vaši platformu vysvětluje, jak funguje obslužná rutina dead-letter a jak ji spouštíte.

Ukázkový program Distribuční seznam

Ukázka distribučního seznamu amqsptl0 poskytuje příklad vložení zprávy do několika front zpráv. Je založena na vzorku MQPUT amqsput0.

Spuštění ukázky distribučního seznamu, amqsptl0

Ukázka Distribuční seznam se spustí podobným způsobem jako ukázky Put.

Je třeba provést následující parametry:

- Názvy front
- Názvy správců front

Tyto hodnoty jsou zadány jako dvojice. Příklad:

```
amqsptl0 queue1 qmanagername1 queue2 qmanagername2
```

Fronty jsou otevírány pomocí MQOPEN a zprávy jsou vloženy do front pomocí příkazu MQPUT. Kódy příčiny jsou vraceny, pokud nejsou rozpoznány některé názvy front nebo správců front.

Nezapomeňte definovat kanály mezi správci front, aby mezi nimi mohly být zprávy toku zpráv. Ukázkový program to pro vás neprovede.

Návrh ukázky distribučního seznamu

umístění záznamů zpráv (MQPMRs) uvádí atributy zpráv pro každé místo určení. Ukázka poskytuje hodnoty pro *MsgId* a *CorrelId*a tyto hodnoty přepíší hodnoty uvedené ve struktuře MQMD.

Pole *PutMsgRecFields* ve struktuře MQPMO označuje, která pole se nacházejí v MQPMRs:

```
  MQLONG PutMsgRecFields=MQPMRF_MSG_ID + MQPMRF_CORREL_ID;
```

Dále ukázka přidělí záznamy odezvy a záznamy objektů. Záznamy objektů (MQORs) vyžadují alespoň jeden pár názvů a sudý počet názvů, to jest, *ObjectName* a *ObjectQMgrName*.

Další fáze zahrnuje připojení ke správcům front pomocí volání MQCONN. Ukázka se pokusí o připojení ke správci front přidruženému k první frontě v MQOR; pokud toto selže, projde záznamy objektů na oplátku. Jste informováni o tom, že není možné připojit se k žádnému správci front a program se ukončí.

Cílové fronty se otevřou pomocí MQOPEN a zpráva je vložena do těchto front pomocí příkazu MQPUT. Všechny problémy a selhání jsou hlášeny v záznamech odpovědí (MQRRs).

Nakonec jsou cílové fronty uzavřeny pomocí funkce MQCLOSE a program se odpojí od správce front pomocí funkce MQDISC. Stejné záznamy odpovědí se používají pro každé volání, které uvádí *CompCode* a *Reason*.

Ukázkové programy Echo

Ukázkové programy Echo echo zprávy z fronty zpráv do fronty odpovědí.

Názvy těchto programů viz "Vlastnosti demonstrovány v ukázkových programech" na stránce 94 .

Programy jsou určeny ke spouštění jako spouštěné programy.

Na systémech UNIX, Linux a Windows je jejich jediným vstupem struktura MQTMC2 (zpráva spouštěče), která obsahuje název cílové fronty a správce front. Verze COBOL používá výchozího správce front.

Pokud jste definici správně nastavili, nejprve spusťte AMQSERV4 v jedné úloze a poté spusťte příkaz AMQSREQ4 v jiném. Můžete použít AMQSTRG4 místo AMQSERV4, ale potenciální zpoždění při odeslání úlohy by mohla méně snadno sledovat, co se děje.

Ukázkové programy požadavku slouží k odesílání zpráv do fronty SYSTEM.SAMPLE.ECHO. Ukázkové programy Echo odešlou zprávu s odpovědí obsahující data ve zprávě požadavku do fronty odpovědi určené ve zprávě s požadavkem.

Návrh ukázkových programů Echo

Program otevře frontu uvedenou ve struktuře zpráv spouštěče, která byla předána, když byla spuštěna. (Pro srozumitelnost zavoláme tuto *frontu požadavků*.) Program používá volání MQOPEN k otevření této fronty pro sdílený vstup.

Program používá volání MQGET k odstranění zpráv z této fronty. Toto volání používá volby MQGMO_ACCEPT_TRUNCATED_MSG, MQGMO_CONVERT a MQGMO_WAIT, s intervalem čekání 5 sekund. Program testuje deskriptor každé zprávy za účelem zjištění, zda se jedná o zprávu požadavku; pokud není, program zahodí zprávu a zobrazí varovnou zprávu.

Pro každý řádek vstupu program pak přečte text do vyrovnávací paměti a pomocí volání MQPUT1 vloží do fronty pro odpověď zprávu obsahující text této řádky.

Pokud se volání MQGET nezdaří, program vloží zprávu s hlášením do fronty odpovědí, přičemž nastaví pole *Feedback* deskriptoru zpráv na kód příčiny vrácený příkazem MQGET.

Pokud ve frontě požadavků nejsou žádné zprávy, program tuto frontu zavře a odpojí se od správce front.

Ukázkové programy Get

Vzorové programy typu Get získají zprávy z fronty pomocí volání MQGET.

Názvy těchto programů viz "Vlastnosti demonstrovány v ukázkových programech" na stránce 94 .

Návrh ukázkového programu Get

Program otevře cílovou frontu pomocí volání MQOPEN s volbou MQOO_INPUT_AS_Q_DEF. Nemůže-li frontu otevřít, zobrazí se v programu chybová zpráva obsahující kód příčiny vrácený voláním MQOPEN.

Pro každou zprávu ve frontě program používá volání MQGET k odebrání zprávy z fronty a poté zobrazí data obsažená ve zprávě. Volání MQGET používá volbu MQGMO_WAIT a určuje *WaitInterval* o 15 sekund, takže program čeká na toto období, pokud ve frontě není žádná zpráva. Pokud před uplynutím tohoto intervalu nepřijde žádná zpráva, volání se nezdaří a vrátí kód příčiny MQRC_NO_MSG_AVAILABLE.

Tento program demonstruje, jak musíte vymazat pole *MsgId* a *CorrelId* struktury MQMD po každém volání MQGET, protože volání nastavuje tato pole na hodnoty obsažené ve zprávě, kterou načítá. Vymazání těchto polí znamená, že po sobě jdoucí příkazy MQGET načítají zprávy v pořadí, ve kterém jsou zprávy zadrženy ve frontě.

Volání MQGET určuje vyrovnávací paměť pevné velikosti. Je-li zpráva delší než tato vyrovnávací paměť, volání selže a program se zastaví.

Program pokračuje, dokud buď volání MQGET nevrátí kód příčiny MQRC_NO_MSG_AVAILABLE, nebo se volání MQGET nezdaří. Pokud se volání nezdaří, zobrazí program chybovou zprávu, která obsahuje kód příčiny.

Program poté zavře frontu pomocí volání MQCLOSE.

Spuštění ukázek amqsget a amqsgetc

Všechny tyto programy mají dva parametry:

1. Název zdrojové fronty (povinné)
2. Název správce front (volitelný)

Není-li správce front zadán, funkce amqsget se připojí k výchozímu správci front a funkce amqsgetc se připojí ke správci front určenému proměnnou prostředí nebo definičním souborem kanálu klienta.

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`.

Pokud vynecháte `qmanagername`, programy předpokládají výchozí nastavení nebo, v případě klienta MQI, správce front identifikovaný proměnnou prostředí nebo definičním souborem kanálu klienta.

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 nakonfigurovat, 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 126.

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 WebSphere 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 WebSphere 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 WebSphere 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 WebSphere MQ MQI client . Kontroluje, zda správce front IBM WebSphere 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 *QueueManager*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.
- *InTransactionInTransaction* zpráv se vkládají do každé transakce. Transakce se opakuje *RepeatCount* krát. 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 *SideQueueNázev* ke kontrole, zda jsou všechny zprávy úspěšně potvrzeny nebo odvolány z produktu *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

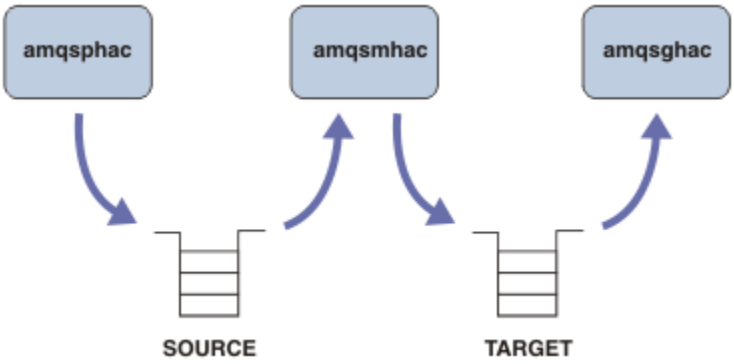
Chcete-li spustit ukázky, musíte nakonfigurovat kanál připojení klienta a serveru. Procedura verifikace klienta vysvětluje, jak nastavit testovací prostředí klienta. Viz Ověření instalace klienta.

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 19 na stránce 126.



Obrázek 19. 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 127 v části “Nastavení a řízení správce front” na stránce 126.)
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** v systému Windows jsou uvedeny v následujících příkladech.

Nastavení a řízení správce front

1. Vytvořte správce front.

```
C:\>crtmqm QM1
WebSphere MQ queue manager created.
Directory 'C:\IBM\MQ\MQ7\Data\qmgrs\QM1' created.
Creating or replacing default objects for QM1.
Default objects statistics : 67 created. 0 replaced. 0 failed.
Completing setup.
Setup completed.
```

   Zapamatujte si datový adresář, abyste nastavili proměnnou **MQCHLLIB** později.

2. Spusťte správce front.

```
C:\>strmqm QM1
WebSphere MQ queue manager 'QM1' starting.
5 log records accessed on queue manager 'QM1' during the log replay phase.
Log replay for queue manager 'QM1' complete.
Transaction manager state recovered for queue manager 'QM1'.
WebSphere MQ queue manager 'QM1' started.
```

3. 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: WebSphere MQ queue created.
     2 : DEFINE QLOCAL(TARGET) REPLACE
AMQ8006: WebSphere MQ queue created.
     3 : DEFINE CHANNEL(CHANNEL1) CHLTYPE(SVRCONN)   TRPTYPE(TCP) MCAUSER(MUSR_MQADMIN)
REPLACE
AMQ8014: WebSphere MQ channel created.
     4 : DEFINE CHANNEL(CHANNEL1) CHLTYPE(CLNTCONN)  TRPTYPE(TCP) CONNAME('LOCALHOST(2345)')
QMNAME(QM1) REPLACE
AMQ8014: WebSphere MQ channel created.
     5 : ALTER LISTENER(SYSTEM.DEFAULT.LISTENER.TCP) TRPTYPE(TCP) PORT(2345)
AMQ8623: WebSphere MQ listener changed.
     6 : START LISTENER(SYSTEM.DEFAULT.LISTENER.TCP)
AMQ8021: Request to start WebSphere MQ Listener accepted.
     7 : START CHANNEL(CHANNEL1)
AMQ8018: Start WebSphere MQ channel accepted.
7 MQSC commands read.
No commands have a syntax error.
All valid MQSC commands were processed.
```

4. Zpřístupněte tabulku kanálů klienta klientům.

   Použijte datový adresář vrácený z příkazu **crtmqm** v kroku “1” na stránce 126a 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.
WebSphere MQ queue manager 'QM1' ending.
WebSphere MQ queue manager 'QM1' ended.

C:\>strmqm QM1
WebSphere MQ queue manager 'QM1' starting.
5 log records accessed on queue manager 'QM1' during the log replay phase.
Log replay for queue manager 'QM1' complete.
Transaction manager state recovered for queue manager 'QM1'.
WebSphere MQ queue manager 'QM1' started.
```

amqsphac

```
Sample AMQSPHAC start
target queue is SOURCE
message <Message 1>
message <Message 2>
16:25:22 : EVENT : Connection Reconnecting (Delay: 0ms)
16:25:45 : EVENT : Connection Reconnecting (Delay: 0ms)
16:26:02 : EVENT : Connection Reconnectedmessage
<Message 3>
```

```
message <Message 4>
message <Message 5>
```

amqsmhac

```
Sample AMQSMHA0 start
16:25:22 : EVENT : Connection Reconnecting (Delay: 0ms)
16:25:45 : EVENT : Connection Reconnecting (Delay: 0ms)
16:26:02 : EVENT : Connection Reconnected
No more messages.
Sample AMQSMHA0 end
C:\>
```

amqsghac

```
Sample AMQSGHAC start
message <Message 1>
message <Message 2>
16:25:22 : EVENT : Connection Reconnecting (Delay: 0ms)
16:25:45 : EVENT : Connection Reconnecting (Delay: 0ms)
16:26:02 : EVENT : Connection Reconnected
message <Message 3>
message <Message 4>
message <Message 5>
```

Související úlohy

Ověření chování sdíleného systému souborů

Související odkazy

amqmfsck (kontrola systému souborů)

Ukázkové programy pro zjišťování

Dotazové ukázkové programy se dotazujeme na některé atributy fronty pomocí volání MQINQ.

Názvy těchto programů viz “Vlastnosti demonstrovány v ukázkových programech” na stránce 94 .

Tyto programy jsou určeny ke spouštění jako spouštěné programy, takže jejich jediným vstupem je struktura MQTMC2 (spouštěcí zpráva) pro systémy IBM i, Windows, UNIX and Linux . Tato struktura obsahuje název cílové fronty s atributy, které mají být zjišťovány. Verze jazyka C také používá název správce front. Verze COBOL používá výchozího správce front.

Chcete-li, aby spouštěcí proces fungoval, ujistěte se, že dotazovací program zjišťování, který chcete použít, je spuštěn zprávami přicházejícími do fronty SYSTEM.SAMPLE.INQ. Chcete-li tak učinit, zadejte název dotazovacího programu pro zjišťování, který chcete použít v poli *ApplicId* definice procesu SYSTEM.SAMPLE.INQPROCESS. Ukázková fronta má typ spouštěče FIRST; pokud již ve frontě existují zprávy, než spustíte ukázku požadavku, ukázka dotazu se nespustí zprávami, které odešlete.

Pokud jste správně nastavili definici:

- Pro systémy UNIX, Linux a Windows spusťte program **runmqtrm** v jedné relaci a pak spusťte program `amqsreq` v jiném.

Ukázkové programy požadavku slouží k odesílání zpráv požadavků, z nichž každá obsahuje pouze název fronty, do fronty SYSTEM.SAMPLE.INQ. Pro každou zprávu požadavku odesílají ukázkové programy Inquire zprávu s informacemi o frontě zadané ve zprávě požadavku. Odpovědi se posílají do fronty odpovědí uvedené ve zprávě s požadavkem.

Návrh ukázkového programu Inquire

Program otevře frontu uvedenou ve struktuře zpráv spouštěče, která byla předána, když byla spuštěna. (Pro srozumitelnost zavoláme tuto *frontu požadavků*.) Program používá volání MQOPEN k otevření této fronty pro sdílený vstup.

Program používá volání MQGET k odstranění zpráv z této fronty. Toto volání používá volby MQGMO_ACCEPT_TRUNCATED_MSG a MQGMO_WAIT, s intervalem čekání 5 sekund. Program testuje deskriptor každé zprávy za účelem zjištění, zda se jedná o zprávu požadavku; pokud není, program zahodí zprávu a zobrazí varovnou zprávu.

Pro každou zprávu požadavku odebranou z fronty požadavků program přečte název fronty (kterou nazýváme *cílovou frontou*) obsaženou v datech a otevře tuto frontu pomocí volání MQOPEN s volbou MQOO_INQ. Program potom použije volání MQINQ k dotazům na hodnoty atributů *InhibitGet*, *CurrentQDepth*a *OpenInputCount* cílové fronty.

Je-li volání MQINQ úspěšné, program použije volání MQPUT1 k vložení zprávy odpovědi do fronty pro odpovědi. Tato zpráva obsahuje hodnoty tří atributů.

Je-li volání MQOPEN nebo MQINQ neúspěšné, program použije volání MQPUT1 k vložení zprávy do fronty do fronty pro odpověď. V poli *Feedback* v deskriptoru zprávy této zprávy je kód příčiny vrácený voláním MQOPEN nebo MQINQ, v závislosti na tom, který z nich selhal.

Po volání MQINQ program zavře cílovou frontu pomocí volání MQCLOSE.

Pokud ve frontě požadavků nejsou žádné zprávy, program tuto frontu zavře a odpojí se od správce front.

The Inquire Properties of a Message Handle sample program

AMQSIQMA je ukázkový program C pro zjištění vlastností obsluhy zprávy z fronty zpráv a je příkladem použití volání rozhraní API MQINQMP.

Tato ukázka vytvoří popisovač zprávy a vloží jej do pole MsgHandle struktury MQGMO. Ukázka pak získá jednu zprávu a vypíše a vytiskne všechny vlastnosti, se kterými byl popisovač zprávy naplněn daty.

```
C:\Program Files\IBM\WebSphere MQ\tools\c\Samples\Bin >amqsiqm Q QM1
Sample AMQSIQMA start
property name <MyProp> value <MyValue>
message text <Hello world!>
Sample AMQSIQMA end
```

Ukázkové programy publikování/odběru

Ukázkové programy publish/subscribe demonstrují použití funkcí publikování a odběru v produktu WebSphere MQ.

K dispozici jsou tři ukázkové programy jazyka C ilustrující způsob programování v rozhraní pro publikování a odběr produktu WebSphere MQ . Existuje několik ukázek jazyka C, které používají starší rozhraní, a existují ukázky Java. Ukázky jazyka Java používají rozhraní WebSphere MQ publish/subscribe v com.ibm.mq.jar a rozhraní JMS publish/subscribe v com.ibm.mqjms. Ukázky rozhraní JMS nejsou zahrnuty v tomto tématu.

C

Najděte ukázku vydavatele `amqspub` ve složce ukázek produktu C . Spusťte jej s libovolným názvem tématu, který chcete použít jako první parametr, následovaný volitelným názvem správce front. Například `amqspub mytopic QM3` . Existuje také verze klienta nazvaná `amqspubc`. Pokud se rozhodnete spustit verzi klienta, nejprve si prohlédněte “Příprava a spuštění ukázkových programů” na stránce 104 , kde získáte podrobnosti.

Vydavatel se připojí k výchozímu správci front a odpoví s výstupem, `target topic is mytopic` . Každý řádek, který zadáte do tohoto okna od této chvíle, bude publikován do `mytopic`.

Otevřete jiné příkazové okno ve stejném adresáři a spusťte program odběratele `amqssub`, dodejte jej se stejným názvem tématu a volitelným názvem správce front. Například `amqssub mytopic QM3`.

Odběratel odpovídá na výstup, `Calling MQGET : 30 seconds wait time`. Od této chvíle se řádky, které zadáte do vydavatele, objevují ve výstupu odběratele.

Spusťte jiného odběratele v jiném okně s příkazovým řádkem a sledujte, jak odběratelé přijímají publikace.

Úplnou dokumentaci parametrů, včetně nastavení voleb, naleznete v ukázkovém zdrojovém kódu. Hodnoty pro pole voleb odběratele jsou popsány v následujícím tématu: Volby (MQLONG).

Existuje další ukázka odběratele amqssbx, která nabízí další volby odběru jako přepínače příkazového řádku.

Zadejte příkaz `amqssbx -d mysub -t mytopic -k` , chcete-li vyvolat odběratele pomocí trvalých odběrů, které jsou uchovány po ukončení odběratele.

Otestujte odběr publikováním další položky s použitím vydavatele. Počkejte 30 sekund, než se odběratel ukončí. Publikujte některé další položky pod stejným tématem. Restartujte odběratele. Poslední položka publikovaná v době, kdy odběratel nebyl spuštěn, ji účastník zobrazí ihned po restartu.

odkaz C

K dispozici je další sada ukázek jazyka C, které demonstrují příkazy ve frontě. Některé z těchto ukázek byly původně odeslány jako součást sady MQ0C Supportpac. Funkce, které ukázky demonstrují, jsou plně podporovány z důvodu kompatibility.

Odrazujeme vás od použití rozhraní příkazového řádku ve frontě. Je mnohem komplexnější než rozhraní API pro publikování/odběr a neexistuje žádný přesvědčivý funkční důvod pro programování složitých příkazů zařazených do fronty. Může se však nacházet vhodnější přístup ve frontě, například proto, že rozhraní již používáte, nebo proto, že vaše programovací prostředí usnadňuje sestavování složité zprávy a volání generického volání MQPUT, místo aby se vytvářela odlišná volání MQSUB.

Další ukázky jsou umístěny v podadresáři `pubsub` ve složce `samples` .

V produktu Tabulka 20 na stránce 130je uvedeno šest typů ukázek.

Tabulka 20. Kategorie starších ukázkových programů jazyka C pro publikování/odběr

Kategorie	Programy	Komentáře
RFH1	`amqssr1a.c` `amqspr1a.c`	Příklad jednoduchého publikování/odběru sestaveného pomocí zpráv ve formátu RFH1 .
RFH2	`amqssr2a.c` `amqspr2a.c`	Příklad jednoduchého publikování/odběru sestaveného pomocí zpráv ve formátu RFH2 .
Ukázky MQAI	`amqsppca.c` `amqsspca.c`	Příklad jednoduchého publikování/odběru sestaveného pomocí příkazů PCF a rozhraní příkazového řádku MQAI.
MA0C Služba výsledků pomocí RFH1	`amqsgama.c` `amqsresa.c`	Služba výsledků vytvořená pomocí záhlaví RFH1 1. Vyžaduje fronty definované v `amqsgama.tst` a `amqsresa.tst` 2. Server `amqsresa` musí být spuštěn před `amqsgama`
MA0C Služba výsledků pomocí RFH2	`amqsgr2a.c` `amqsrr2a.c`	Služba výsledků sestavená pomocí záhlaví RFH2 1. Vyžaduje fronty definované v `amqsgama.tst` a `amqsresa.tst` 2. Server `amqsresa` musí být spuštěn před `amqsgama`

Tabulka 20. Kategorie starších ukázkových programů jazyka C pro publikování/odběr (pokračování)		
Kategorie	**Programy**	**Komentáře**
Ukázka uživatelské procedury pro publikování/odběr směrování	`amqspsra.c`	Demonstruje, jak změnit místo určení fronty nebo správce front pro zprávu publikování/odběru ve výstupním rámci směrování.

Java

Ukázka Java `MQPubSubApiSample.java` kombinuje vydavatele a odběratele v jednom programu. Jeho zdrojový a kompilovaný soubor třídy se nacházejí ve složce ukázek produktu `wmqjava` .

Pokud se rozhodnete spustit v režimu klienta, nejprve si prohlédněte “Příprava a spuštění ukázkových programů” na stránce 104 , kde získáte podrobnosti.

Spusťte ukázku z příkazového řádku pomocí příkazu Java, máte-li nakonfigurované prostředí Java. Můžete také spustit ukázku z pracovního prostoru produktu WebSphere MQ Explorer Eclipse , který má již nastavovací pracovní plochu pro programování v jazyce Java.

Možná budete muset změnit některé vlastnosti ukázkového programu, abyste jej mohli spustit. To provedete tak, že poskytnete parametry do prostředí JVM, nebo upravíte zdroj.

Pokyny v produktu “Spuštění ukázkyApiSample jazyka Java MQPubSub” na stránce 131 ukazují, jak spustit ukázku z pracovního prostoru Eclipse .

Spuštění ukázkyApiSample jazyka Java MQPubSub

Postup při spuštění produktu MQPubSubApiSample pomocí nástrojů Java Development Tools z platformy Eclipse .

Než začnete

Otevřete pracovní plochu Eclipse . Vytvořte nový adresář pracovního prostoru a vyberte jej. Zavřete uvítací okno.

Postupujte podle kroků v produktu “Příprava a spuštění ukázkových programů” na stránce 104 před spuštěním jako klienta.

Informace o této úloze

Ukázkový program pro publikování/odběr jazyka Java je program jazyka Java klienta WebSphere MQ MQI. Ukázka se spustí bez úprav s použitím výchozího správce front, který naslouchá na portu 1414. Úloha popisuje tento jednoduchý případ a obecně označuje, jak poskytnout parametry, a upravit ukázku tak, aby odpovídala různým konfiguracím produktu WebSphere MQ . Příklad je ilustrován na systému Windows. Cesty k souborům se budou lišit i na jiných platformách.

Postup

1. Import ukázkových programů v jazyce Java
   a) Na pracovní ploše klepněte na volbu **Okno** > **Otevřít perspektivu** > **Další** > **Java** a klepněte na tlačítko **OK**.
   b) Přepněte do pohledu **Průzkumník balíků** .
   c) Klepněte pravým tlačítkem myši v zobrazení **Průzkumník balíků** do seznamu povolených položek. Klepněte na volbu **Nový** > **Projekt Java**.

d) V poli **Project name** napište `MQ Java Samples`. Klepněte na tlačítko **Další**.

e) V panelu **Java Settings** přepněte na kartu **Knihovny** .

f) Klepněte na volbu **Přidat externí soubory JAR**.

g) Přejděte do adresáře *MQ_INSTALLATION_PATH*`\java\lib` , kde *MQ_INSTALLATION_PATH* je instalační složka produktu WebSphere MQ a vyberte volbu `com.ibm.mq.jar` a `com.ibm.mq.jmqi.jar` .

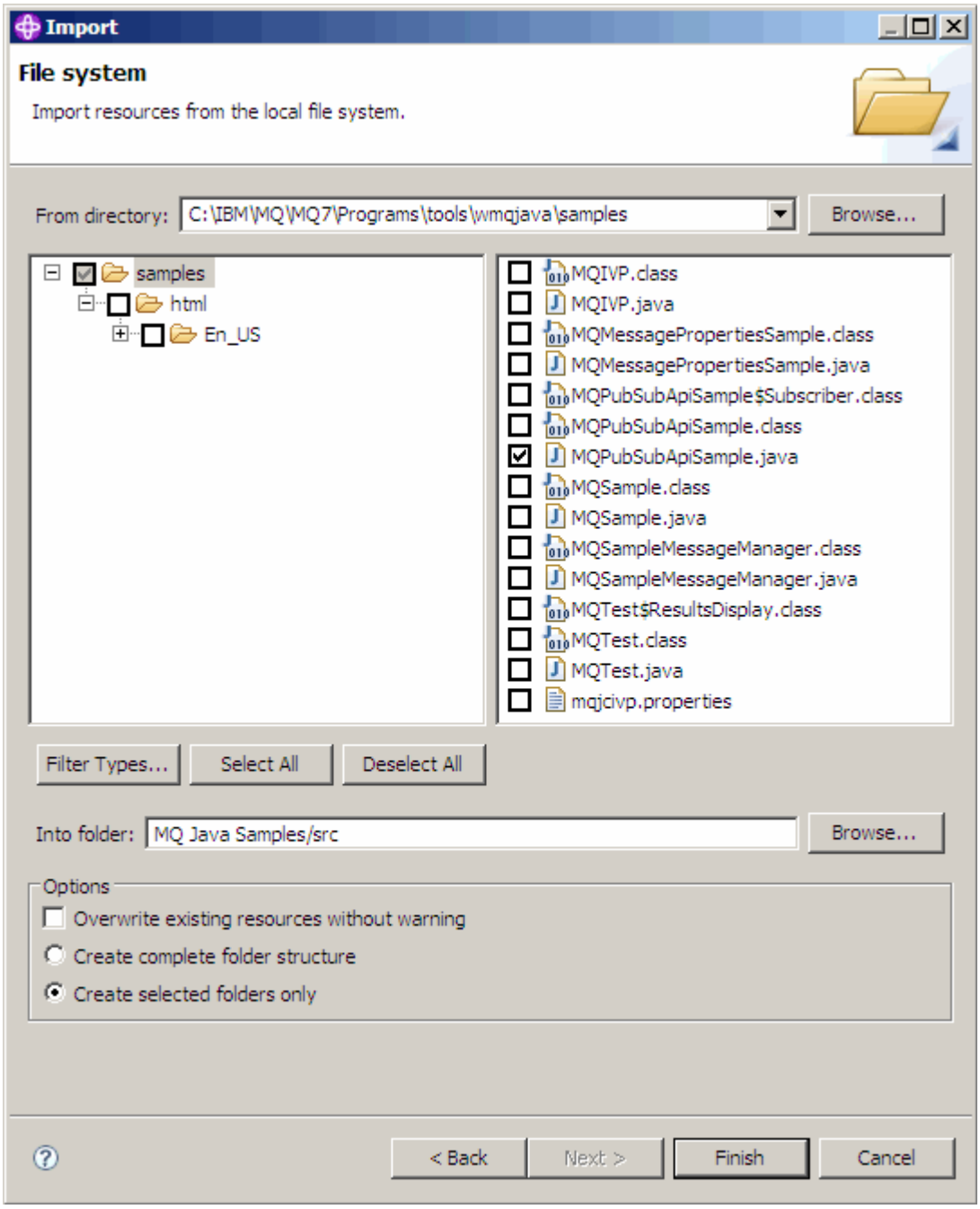
h) Klepněte na volbu **Otevřít** > **Dokončit.**

i) Klepněte pravým tlačítkem myši na položku `src` v zobrazení **Průzkumník balíků** .

j) Vyberte **Importovat ...** > **Obecné** > **Systém souborů** > **Další** > **Procházet**... a přejděte k cestě *MQ_INSTALLATION_PATH*`\tools\wmqjava\samples` , kde *MQ_INSTALLATION_PATH* je instalační adresář produktu WebSphere MQ .

k) Na panelu **Import** produktu Obrázek 20 na stránce 133klepněte na `samples` (nezaškrtujte zaškrtávací políčko).

l) Vyberte volbu `MQPubSubApiSample.java`. Pole **Into folder** by mělo obsahovat `MQ Java Samples/src`. Klepněte na tlačítko **Dokončit**.



Obrázek 20. Import systému souborů

2. Spusťte ukázkový program pro publikování/odběr.

   Existují dva způsoby spuštění programu, v závislosti na tom, zda je třeba změnit výchozí parametry.

   - První volba spustí program bez provedení jakýchkoli změn:
     - V hlavní nabídce pracovního prostoru rozbalte složku `src` . Right-click **MQPubSubApiSample.javaSpustit jako** > **1. Aplikace Java**
   - Druhá volba spustí program s parametry nebo s modifikovaným zdrojovým kódem pro vaše prostředí:
     - Otevřete produkt `MQPubSubApiSample.java` a prostudujte konstruktor produktu `MQPubSubApiSample` .

– Upravte atributy programu.

  Tyto atributy lze měnit pomocí přepínače -D JVM nebo zadáním výchozí hodnoty pro systémovou vlastnost úpravou zdrojového kódu.

  - topicObject
  - QueueManagerName
  - subscriberCount

  Tyto atributy lze měnit pouze úpravou zdrojového kódu v konstruktoru.

  - hostname
  - Port
  - kanál

  Chcete-li nastavit vlastnosti systému, zadejte v přístupovém mechanizmu výchozí hodnotu, například:

```
queueManagerName = System.getProperty("com.ibm.mq.pubSubSample.queueManagerName",
"QM3");
```

  Případně uveďte parametr do prostředí JVM pomocí volby `-D` , jak ukazuje následující postup:

a. Zkopírujte úplný název souboru System.Property , který chcete nastavit, například: `com.ibm.mq.pubSubSample.queueManagerName`.
b. V pracovním prostoru klepněte pravým tlačítkem myši na volbu **Spustit** > **Otevřít dialogové okno Spustit**. Poklepejte na aplikaci Java v sekci **Vytvořit, spravovat a spustit aplikace** a klepněte na kartu **(x) = Argumenty** .
c. V podokně **Argumenty virtuálního počítače:** zadejte příkaz `-D` a vložte název System.property , `com.ibm.mq.pubSubSample.queueManagerName`, následovaný hodnotou `=QM3`. Klepněte na volbu **Použít** > **Spustit**.
d. Přidejte další argumenty jako seznam oddělený čárkami, nebo jako další řádky v podokně, bez oddělovačů.

   Například:`-Dcom.ibm.mq.pubSubSample.queueManagerName=QM3`, `-Dcom.ibm.mq.pubSubSample.subscriberCount=6`.

Ukázkový program publikování a ukončení

Příkaz AMQSPSE0 je ukázkový program jazyka C pro proceduru ukončení publikování, než je doručena odběrateli. Ukončení může například změnit záhlaví zprávy, informační obsah nebo místo určení, nebo zabránit publikování zprávy na odběrateli.

Chcete-li spustit ukázku, proveďte následující úlohy:

1. Konfigurujte správce front:
   - V systémech UNIX and Linux přidejte oddíl podobný tomuto souboru `qm.ini` :

```
PublishSubscribe:
           PublishExitPath=<Module>
           PublishExitFunction=EntryPoint
```

   kde je modul *MQ_INSTALLATION_PATH*`/samp/bin/amqspse` . *MQ_INSTALLATION_PATH* Představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován. V systému Windows nastavte ekvivalentní atributy v registru.
2. Ujistěte se, že je modul přístupný pro produkt WebSphere MQ.
3. Restartujte správce front, abyste vybrali konfiguraci.
4. V procesu aplikace, který má být trasován, popište, kam mají být trasovací soubory zapisovány. Příklad:

- V systému UNIX and Linux se ujistěte, že adresář `/var/mqm/trace` existuje a vyexportujte následující proměnnou prostředí:

```
export MQPSE_TRACE_LOGFILE=/var/mqm/trace/PubTrace
```

- V systému Windows se ujistěte, že existuje adresář `C:\temp` a nastavte následující proměnnou prostředí:

```
set MQPSE_TRACE_LOGFILE=C:\temp\PubTrace
```

Ukázkové programy Put

Ukázkové programy vkládání vloží zprávy do fronty pomocí volání MQPUT.

Názvy těchto programů viz “Vlastnosti demonstrovány v ukázkových programech” na stránce 94 .

Návrh ukázkového programu Put

Program používá volání MQOPEN s volbou MQOO_OUTPUT k otevření cílové fronty pro vkládání zpráv.

Pokud nemůže frontu otevřít, výstupem programu je chybová zpráva obsahující kód příčiny vrácený voláním MQOPEN. Chcete-li program ponechat jednoduchý, a to i při následných voláních MQI, program použije výchozí hodnoty pro celou řadu voleb.

Pro každý řádek vstupu program přečte text do vyrovnávací paměti a použije volání MQPUT k vytvoření datagramové zprávy obsahující text této řádky. Program pokračuje, dokud nedojde k dosažení konce vstupu nebo volání MQPUT selže. Pokud se program dostane na konec vstupu, zavře frontu pomocí volání MQCLOSE.

Spuštění ukázkových programů pro vkládání

Spuštění ukázek amqsput a amqsputc

Všechny tyto programy používají 2 parametry:

1. Název cílové fronty (povinné)
2. Název správce front (volitelný)

Není-li určen správce front, připojuje se hodnota amqsput k výchozímu správci front a amqsputc se připojí ke správci front určenému proměnnou prostředí nebo definičním souborem kanálu klienta. Chcete-li spustit tyto programy, zadejte jednu z následujících možností:

- `amqsput myqueue qmanagername`
- `amqsputc myqueue qmanagername`

kde `myqueue` je název fronty, na kterou se mají zprávy vložit, a `qmanagername` je správce front, který vlastní `myqueue`.

Spuštění ukázky amq0put

Verze COBOL nemá žádné parametry. Připojuje se k výchozímu správci front, a když jej spustíte, budete vyzváni:

```
Please enter the name of the target queue
```

Vezme vstup z StdIn a přidá každý řádek vstupu do cílové fronty. Prázdný řádek označuje, že již nejsou žádná data.

Ukázkové programy Referenční zprávy

Ukázky referenční zprávy umožňují přenos velkého objektu z jednoho uzlu do jiného (obvykle na různých systémech) bez nutnosti uložení objektu do front WebSphere MQ ve zdroji nebo v cílových uzlech.

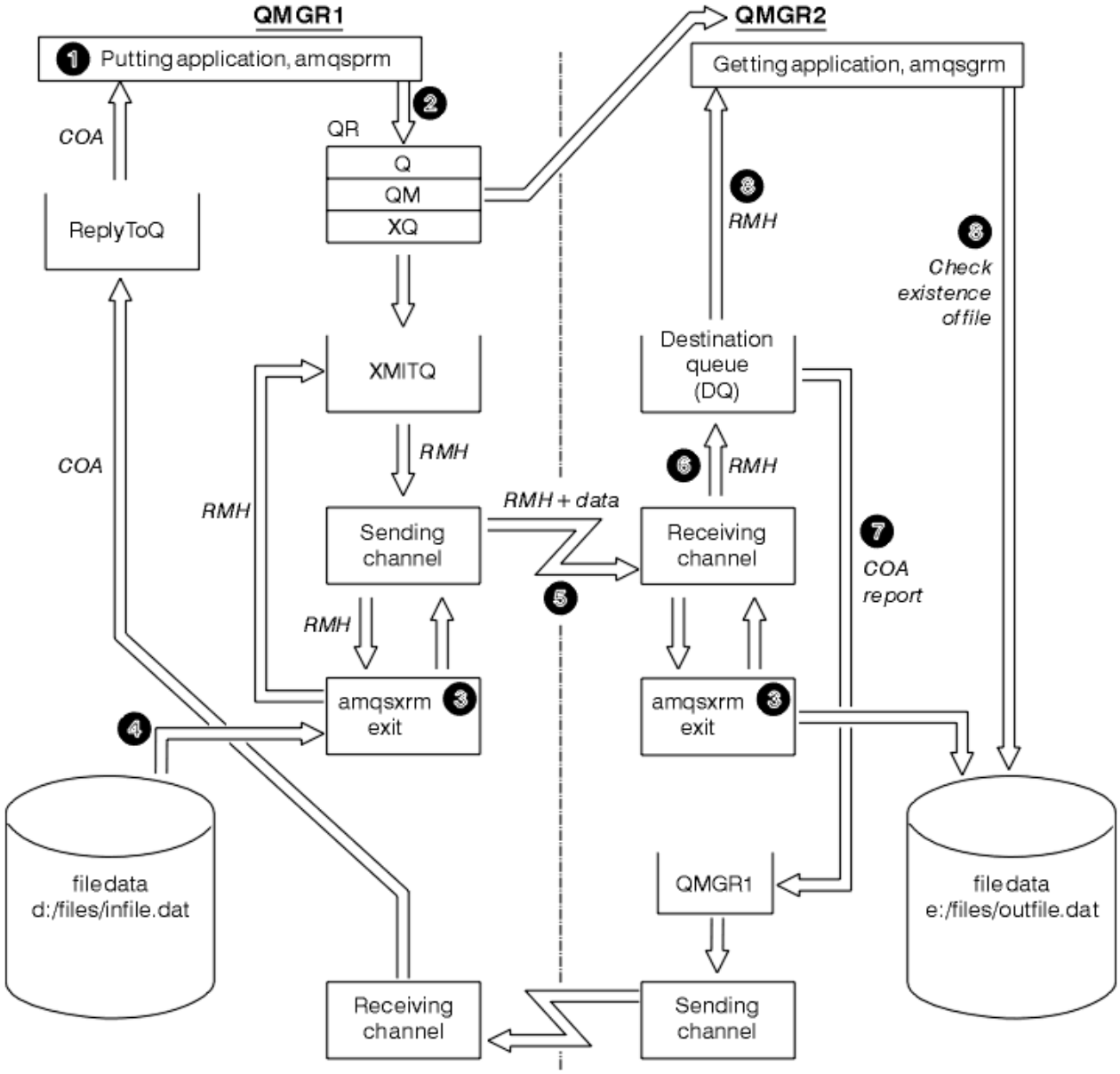
K dispozici je sada ukázkových programů, které demonstrují, jak lze do fronty vložit referenční zprávy, které jsou přijímány uživatelskými procedurami a které byly převzaty z fronty. Vzorové programy používají referenční zprávy k přesunu souborů. Chcete-li přesunout jiné objekty, jako jsou databáze, nebo chcete-li provést kontroly zabezpečení, definujte vlastní ukončení založené na našem vzorku amqsxrm. V následujících oddílech jsou popsány ukázkové programy Referenční zprávy.

Verze ukázkového programu výstupního bodu zpráv, který má být použit, závisí na platformě, na které je kanál spuštěn. Na všech platformách použijte na odesílající straně amqsxrma. Použijte amqsxrma na přijímajícím konci, pokud je příjemce spuštěn v libovolném produktu WebSphere MQ s výjimkou produktu WebSphere MQ pro produkt IBM i.

Spuštění ukázek referenční zprávy

V této části se dozvíte, jak spustit ukázkové programy Referenční zprávy.

Ukázky referenční zprávy se spouští následujícím způsobem:



Obrázek 21. Spuštění ukázek referenční zprávy

1. Nastavte prostředí pro spuštění listenerů, kanálů a monitorů spouštěčů a definujte kanály a fronty.

   Pro účely popisu způsobu nastavení referenčního příkladu zpráv se tento příklad odkazuje na odesílající stroj jako MACHINE1 se správcem front s názvem QMGR1 a přijímajícím počítačem jako MACHINE2 s použitím správce front nazvaného QMGR2.

   Poznámka: Následující definice umožňují sestavení Referenční zprávy k odeslání souboru s typem objektu FLATFILE ze správce front QMGR1 do QMGR2 a k opětného vytvoření souboru, jak je definován ve volání do AMQSPRM (nebo AMQSPRMA na systému IBM i). Referenční zpráva (včetně dat souboru) se odešle pomocí kanálu CHL1 a přenosové fronty XMITQ a umístí se do fronty DQ. Výjimka a zprávy COA jsou odeslány zpět do QMGR1 pomocí zprávy kanálu REPORT a přenosové fronty QMGR1.

   Aplikace, která přijme referenční zprávu (AMQSGRM), 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 kde je nainstalován produkt WebSphere MQ .

   Definice MQSC používají styl AIX pro definování uživatelských procedur. Je důležité si uvědomit, že data zprávy FLATFILE jsou citlivá na velikost písmen a ukázka nebude fungovat, pokud není psána velkými písmeny.

   Na počítači MACHINE1, správce front QMGR1

   Syntaxe MQSC

```
define chl(chl1) chltype(sdr) trptype(tcp) conname('machine2') xmitq(xmitq)
msgdata(FLATFILE) msgexit('/usr/lpp/mqm/samp/bin/amqsxrm(MsgExit)
')

define ql(xmitq) usage(xmitq)

define chl(report) chltype(rcvr) trptype(tcp) replace

define qr(qr) rname(dq) rqmname(qmgr2) xmitq(xmitq) replace
```

   Poznámka: Nezadáte-li název správce front, systém použije výchozího správce front.

```
CRTMQMCHL  CHLNAME(CHL1) CHLTYPE(*SDR) MQMNAME(QMGR1) +
             REPLACE(*YES) TRPTYPE(*TCP) +
             CONNAME('MACHINE2(60501)') TMQNAME(XMITQ) +
             MSGEXIT(QMQM/AMQSXRM4) MSGUSRDATA(FLATFILE)

CRTMQMQ    QNAME(XMITQ) QTYPE(*LCL) MQMNAME(QMGR1) +
             REPLACE(*YES) USAGE(*TMQ)

CRTMQMCHL  CHLNAME(REPORT) CHLTYPE(*RCVR) +
               MQMNAME(QMGR1) REPLACE(*YES) TRPTYPE(*TCP)

CRTMQMQ    QNAME(QR) QTYPE(*RMT) MQMNAME(QMGR1) +
             REPLACE(*YES) RMTQNAME(DQ) +
             RMTMQMNAME(QMGR2) TMQNAME(XMITQ)
```

   Na počítači MACHINE2, správce front QMGR2

   Syntaxe MQSC

```
define chl(chl1) chltype(rcvr) trptype(tcp)
msgexit('/usr/lpp/mqm/samp/bin/amqsxrm(MsgExit)')
        msgdata(flatfile)

define chl(report) chltype(sdr) trptype(tcp) conname('MACHINE1')
        xmitq(qmgr1)

define ql(initq)

define ql(qmgr1) usage(xmitq)

define pro(proc) applicid('/usr/lpp/mqm/samp/bin/amqsgrm')

define ql(dq) initq(initq) process(proc) trigger trigtype(first)
```

2. Po vytvoření objektů produktu WebSphere MQ :
   a. Kde je to vhodné pro platformu, spusťte modul listener pro odesílající a přijímající správce front.
   b. Spuštění kanálů CHL1 a REPORT
   c. Na přijímajícím správci front spusťte monitor spouštěčů pro inicializační frontu INITQ
3. Vyvolejte ukázkový program put referenční zprávy AMQSPRM z příkazového řádku pomocí následujících parametrů:

   -m Název lokálního správce front. Výchozí hodnota je výchozí správce front.

   -i Název a umístění zdrojového souboru

   -o Název a umístění cílového souboru

   -q Název fronty

   -g Název správce front, ve kterém je fronta definovaná v parametru -q, tato výchozí hodnota je určena správci front uvedenému v parametru -m

   -t Typ objektu

   -w Interval čekání, tj. čekací doba pro výjimky a sestavy COA z přijímajícího správce front

   Chcete-li například použít ukázku s dříve definovanými objekty, použijte následující parametry:

```
-mQMGR1 -iInput File -oOutput File -qQR -tFLATFILE -w120
```

   Zvýšení čekací doby umožňuje odeslání času velkého souboru přes síť před tím, než program položení zprávy vyprší.

```
amqsprm -q QR -m QMGR1 -i d:\x\file.in -o d:\y\file.out -t FLATFILE
```

   Poznámka: Pro platformy UNIX and Linux musíte použít dvě zpětná lomítka (\\) místo toho, abyste označili adresář cílového souboru. Proto příkaz **amqsprm** vypadá takto:

```
amqsprm -i /files/infile.dat -o e:\\files\\outfile.dat -q QR
-m QMGR1 -w 30 -t FLATFILE
```

   Spuštění programu pro práci s referenčními zprávami provede následující akce:

   - Referenční zpráva je vložena do fronty QR ve správci front QMGR1.
   - Zdrojový soubor a cesta jsou `d:\files\infile.dat` a existují v systému, kde je vydán příklad příkazu.
   - Je-li fronta QR vzdálenou frontou, odešle se referenční zpráva jinému správci front, na jiném systému, kde je soubor vytvořen s názvem a cestou `e:\files\outfile.dat`. Obsah tohoto souboru je stejný jako zdrojový soubor.
   - amqsprm čeká 30 sekund na zprávu COA z cílového správce front.
   - Typ objektu je `flatfile`, takže kanál použitý pro přesouvání zpráv z fronty QR fronty musí uvádět toto v poli *MsgData* .
4. Definujete-li kanály, vyberte uživatelskou proceduru zprávy v odesílajícím i přijímajícím rámci tak, aby byla amqsxrm. Tato hodnota je definována v produktu WebSphere MQ for Windows takto:

```
msgexit('pathname\amqsxrm.dll(MsgExit)')
```

Tato volba je definována v produktu WebSphere MQ for AIX, WebSphere MQ for HP-UXa WebSphere MQ for Solaris, jak je uvedeno níže:

```
msgexit('pathname/amqsxrm(MsgExit)')
```

Uvedete-li název cesty, zadejte úplný název. Pokud název cesty vynecháte, předpokládá se, že se program nachází v cestě určené v souboru `qm.ini` (nebo v adresáři WebSphere MQ for Windows, cesta uvedená v registru).

5. Uživatelská procedura kanálu přečte záhlaví referenční zprávy a vyhledá soubor, na který odkazuje.
6. Uživatelská procedura kanálu pak může soubor rozdělit před odesláním kanálu spolu se záhlavím. V produktu WebSphere MQ for AIX, WebSphere MQ for HP-UXa WebSphere MQ for Solaris změňte vlastníka skupiny cílového adresáře na hodnotu 'mqm', aby mohla vzorová uživatelská procedura pro zprávy vytvořit soubor v tomto adresáři. Také změňte oprávnění k cílovému adresáři tak, aby do ní mohli zapisovat členové skupiny mqm. Data souboru nejsou uložena ve frontách produktu WebSphere MQ .
7. Když je poslední segment souboru zpracováván uživatelskou procedurou příjmu zpráv, umístí se referenční zpráva do cílové fronty určené parametrem amqsprm. Je-li tato fronta spuštěna (to znamená, že definice uvádí atributy fronty *Trigger*, *InitQ*a *Process*), spustí se program určený parametrem PROC cílové fronty. Program, který má být spuštěn, musí být definován v poli *ApplId* atributu *Process* .
8. Když se referenční zpráva dostane do cílové fronty (DQ), odešle se zpráva COA zpět do aplikace pro uvedení aplikace (amqsprm).
9. Ukázka Získat referenční zprávu (Get Reference Message), amqsgrm, získává zprávy z fronty zadané ve zprávě spouštěcího impulsu a kontroluje existenci souboru.

Návrh ukázky Vložit referenční zprávu (amqsprma.c, AMQSPRM4)

Toto téma poskytuje podrobný popis ukázky Vložit referenční zprávu.

Tato ukázka vytvoří referenční zprávu, která odkazuje na soubor a vkládá ji do zadané fronty:

1. Ukázka se připojí k lokálnímu správci front pomocí příkazu MQCONN.
2. Poté se otevře (MQOPEN) modelové fronty, která se používá k přijímání zpráv sestav.
3. Ukázka sestaví referenční zprávu obsahující hodnoty vyžadované k přesunu souboru, například názvy zdrojového a cílového souboru a typ objektu. Jako příklad, ukázka dodávaná s produktem WebSphere MQ sestaví referenční zprávu pro odeslání souboru `d:\x\file.in` z `QMGR1` na `QMGR2` a pro opětovné vytvoření souboru jako `d:\y\file.out` za použití 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: Pro platformy UNIX and Linux použijte místo jednoho k označení adresáře cílového souboru dvě zpětná lomítka (\\) místo jednoho. Proto příkaz **amqsprm** vypadá takto:

```
amqsprm -q QR -m QMGR1 -i /x/file.in -o d:\\y\\file.out -t FLATFILE
```

4. Referenční zpráva je vložena (bez jakýchkoliv dat souboru) do fronty zadané parametrem /q. Jedná-li se o vzdálenou frontu, je zpráva vložena do příslušné přenosové fronty.
5. Ukázka čeká, po dobu uvedenou v parametru /w (což je výchozí hodnota 15 sekund), pro sestavy COA, které jsou spolu s hlášeními výjimek odeslány zpět do dynamické fronty vytvořené v lokálním správci front (QMGR1).

Návrh ukázky ukončení referenční zprávy (amqsxrma.c, AMQSXRM4)

Tato ukázka rozpoznává referenční zprávy s typem objektu, který odpovídá typu objektu v poli uživatelských dat uživatelské procedury pro zpracování zprávy definice kanálu.

Pro tyto zprávy nastane následující situace:

- Na odesílacím nebo serverovém kanálu se uvedená délka dat zkopíruje z uvedeného posunutí uvedeného souboru do prostoru zbývajícího ve vyrovnávací paměti agenta po referenční zprávě. Není-li dosaženo konce souboru, vrátí se referenční zpráva po aktualizaci pole *DataLogicalOffset* do přenosové fronty.
- V případě žadatele nebo kanálu příjemce, je-li pole *DataLogicalOffset* nula a uvedený soubor neexistuje, je vytvořen. Data následující za referenční zprávou se přidají na konec uvedeného souboru. Není-li referenční zpráva poslední pro uvedený soubor, bude vyřazena. Jinak se vrátí do uživatelské procedury kanálu bez připojených dat, které mají být vloženy do cílové fronty.

Pokud je pro odesílací a serverové kanály pole *DataLogicalLength* ve vstupní referenční zprávě nula, zbývající část souboru, od *DataLogicalOffset* do konce souboru, se má odeslat spolu s kanálem. Pokud není nula, odešle se pouze uvedená délka.

Dojde-li k chybě (například v případě, že ukázka nemůže otevřít soubor), MQCXP.*ExitResponse* je nastaven na MQXCC_SUPPRESS_FUNCTION, aby zpracovávaná zpráva byla vložena do fronty nedoručených zpráv místo toho, aby pokračovala v cílové frontě. Kód zpětné vazby je vrácen v MQCXP.*Feedback* a vrátí se do aplikace, která vložila zprávu do pole *Feedback* v deskriptoru zpráv ve zprávě sestavy. Důvodem je to, že vložení aplikace požadovala výjimku nastavením MQRO_EXCEPTION v poli *Report* MQMD.

Je-li kódování nebo *CodedCharacterSetId* (CCSID) referenční zprávy odlišné od kódování ve správci front, je referenční zpráva převedena na lokální kódování a CCSID. V našem vzorku, amqsprm, formát objektu je MQFMT_STRING, takže amqsxrm převádí data objektu na lokální CCSID na přijímajícím konci před tím, než jsou data zapsána do souboru.

Neuvádějte formát přenášený soubor jako MQFMT_STRING, pokud soubor obsahuje vícebajtové znaky (například DBCS nebo Unicode). Důvodem je skutečnost, že vícebajtový znak lze rozdělit, je-li soubor segmentován na konci odesílání. Chcete-li přenést a převést takový soubor, zadejte formát jako něco jiného než MQFMT_STRING tak, aby se jeho uživatelská procedura pro referenční zprávy nekonvertoval a nepřeváděla soubor na přijímajícím konci po dokončení přenosu.

Kompilace ukázky Ukončení referenční zprávy

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

Chcete-li kompilovat amqsxrma, použijte následující příkazy:

Na systému AIX

```
xlc_r -q64 -e MsgExit -bE:amqsxrm.exp -bM:SRE -o amqsxrm_64_r
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib64 -lmqm_r amqsqrma.c
```

V systému HP-UX

```
$ c89 +DD64 +z -c -D_HPUX_SOURCE -o amqsxrma.o amqsqrma.c -IMQ_INSTALLATION_PATH/inc
$ ld -b amqsxrma.o -o /var/mqm/exits64/amqsxrma -LMQ_INSTALLATION_PATH/lib64
-L/usr/lib/pa20_64 -lmqm_r -lpthread
```

zapLinux

```
$ gcc -m64 -shared -fPIC -o /var/mqm/exits64/amqsxrma amqsqrma.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -lmqm_r
```

V systému Solaris

```
$ cc -xarch=v9 -mt -G -o /var/mqm/exits64/amqsxrma amqsqrma.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -lmqm
```

```
 -lsocket
-lnsl -ldl
```

V systému Windows

Produkt WebSphere MQ nyní dodává knihovnu mqm s klientskými balíky a s balíky serveru, takže následující příklad používá produkt `mqm.lib` namísto produktu `mqmvx.lib`:

```
cl amqsqrma.c /link /out:amqsxrm.dll /dll mqm.lib mqm.lib /def:amqsxrm.def
```

Obecné informace o zápisu a kompilaci kanálů kanálu naleznete v tématu “Psaní programů výstupních bodů kanálu” na stránce 383 .

Návrh ukázky Get Reference Message (amqsgrma.c, AMQSGRM4)

Toto téma vysvětluje návrh ukázky Získat referenční zprávu.

Logika programu je následující:

1. Ukázka je spuštěna a extrahuje názvy front a správců front ze vstupní zprávy spouštěče.
2. Poté se připojí k zadanému správci front pomocí příkazu MQCONN a otevře určenou frontu pomocí příkazu MQOPEN.
3. Ukázka vydá příkaz MQGET s intervalem čekání 15 sekund uvnitř cyklu pro získání zpráv z fronty.
4. Je-li zpráva referenční zprávou, ukázka zkontroluje existenci souboru, který byl přenesen.
5. Poté frontu zavře a odpojí se od správce front.

Ukázkové programy požadavku

Vzorové programy požadavku demonstrují zpracování typu klient/server. Ukázky jsou klienti, kteří vloží zprávy požadavků do cílové fronty serveru, která je zpracována programem serveru. Čekají na program serveru, aby umístili zprávu odpovědi do fronty pro odpověď.

Ukázky požadavků umístili řadu zpráv požadavků do fronty cílového serveru pomocí volání MQPUT. Tyto zprávy určují lokální frontu SYSTEM.SAMPLE.REPLY jako fronta odpovědí, která může být lokální nebo vzdálená fronta. Programy čekají na zprávy odpovědí a pak je zobrazí. Odpovědi se odesílají pouze v případě, že je fronta cílových serverů zpracovávána serverovou aplikací nebo pokud je spuštěna aplikace pro tento účel (jsou navrženy vzorové programy Inquire, Set a Echo). Ukázka C čeká 1 minutu (ukázka COBOL čeká 5 minut), pro první odpověď na příchod (aby umožnil spuštění serverové aplikace) a 15 sekund pro následné odpovědi, ale oba vzorky mohou skončit, aniž by byly obdrženy odpovědi. Názvy ukázkových programů požadavku viz “Vlastnosti demonstrovány v ukázkových programech” na stránce 94 .

Spuštění ukázkových programů požadavku

Spuštění ukázek amqsreq0.c, amqsreq a amqsreqc

Verze C programu má tři parametry:

1. Název fronty cílového serveru (nezbytné)
2. Název správce front (volitelný)
3. Fronta odpovědí (volitelná)

Zadejte například jednu z následujících možností:

- `amqsreq myqueue qmanagername replyqueue`
- `amqsreqc myqueue qmanagername`
- `amq0req0 myqueue`

kde `myqueue` je název fronty cílového serveru, `qmanagername` je název správce front, který vlastní `myqueue`, a `replyqueue` je jméno fronty odpovědí.

Pokud název správce front vynecháte, předpokládá se, že výchozí správce front vlastní tuto frontu. Pokud vynecháte název fronty odpovědí, je poskytnuta výchozí fronta odpovědí.

Spuštění ukázky amq0req0.cbl

Verze COBOL nemá žádné parametry. Připojuje se k výchozímu správci front, a když jej spustíte, budete vyzváni:

```
Please enter the name of the target server queue
```

Program vezme svůj vstup z StdIn a přidá každý řádek do cílové fronty serveru, přičemž každý řádek textu bude mít obsah jako obsah zprávy požadavku. Program skončí, když se čte řádek s hodnotou null.

Spuštění ukázky AMQSREQ4

Program v jazyku C vytváří zprávy tak, že přebírá data ze stdin (klávesnice) s prázdným časem ukončujícím vstup. Program přijímá až tři parametry: název cílové fronty (povinné), název správce front (volitelné) a název fronty pro odpovědi (volitelné). Není-li zadán žádný název správce front, použije se výchozí správce front. Není-li zadána žádná fronta pro odpověď, je uveden parametr SYSTEM.SAMPLE.REPLY se používá.

Zde je příklad, jak volat vzorový program v jazyce C, který uvádí frontu pro odpověď, ale nechání výchozí nastavení správce front:

```
CALL PGM(QMQM/AMQSREQ4) PARM('SYSTEM.SAMPLE.LOCAL' '' 'SYSTEM.SAMPLE.REPLY')
```

Poznámka: Nezapomeňte, že názvy front jsou citlivé na velikost písmen. Všechny fronty vytvořené vzorovým souborem vytvoření AMQSAMP4 mají názvy vytvořené velkými písmeny.

Spuštění ukázky AMQ0REQ4

Program v jazyce COBOL vytváří zprávy tak, že přijímá data z klávesnice. Chcete-li spustit program, zavolejte program a uveďte název cílové fronty jako parametr. Program přijímá vstup z klávesnice do vyrovnávací paměti a vytváří zprávu požadavku pro každý řádek textu. Program se zastaví, když zadáte prázdný řádek na klávesnici.

Spuštění ukázky požadavku pomocí spouštěče

Pokud se ukázka používá se spouštěním a jedním z ukázkových programů Inquire, Set nebo Echo, musí být řádek vstupu název fronty fronty, ke které má spuštěný program přistupovat.

Systémy UNIX, Linux a Windows

Spuštění ukázek pomocí spouštěče:

1. Spusťte program pro monitorování spouštěčů RUNMQTRM v jedné relaci (inicializační frontu SYSTEM.SAMPLE.TRIGGER je k dispozici pro použití).
2. Spusťte program amqsreq v jiné relaci.
3. Ujistěte se, že jste definovali cílovou frontu serveru.

   K dispozici jsou ukázkové fronty, které můžete použít jako frontu cílového serveru pro ukázku požadavku na vložení zpráv:

   - SYSTEM.SAMPLE.INQ -pro ukázkový program Inquire
   - SYSTEM.SAMPLE.SET -pro ukázkový program Set
   - SYSTEM.SAMPLE.ECHO -pro ukázkový program Echo

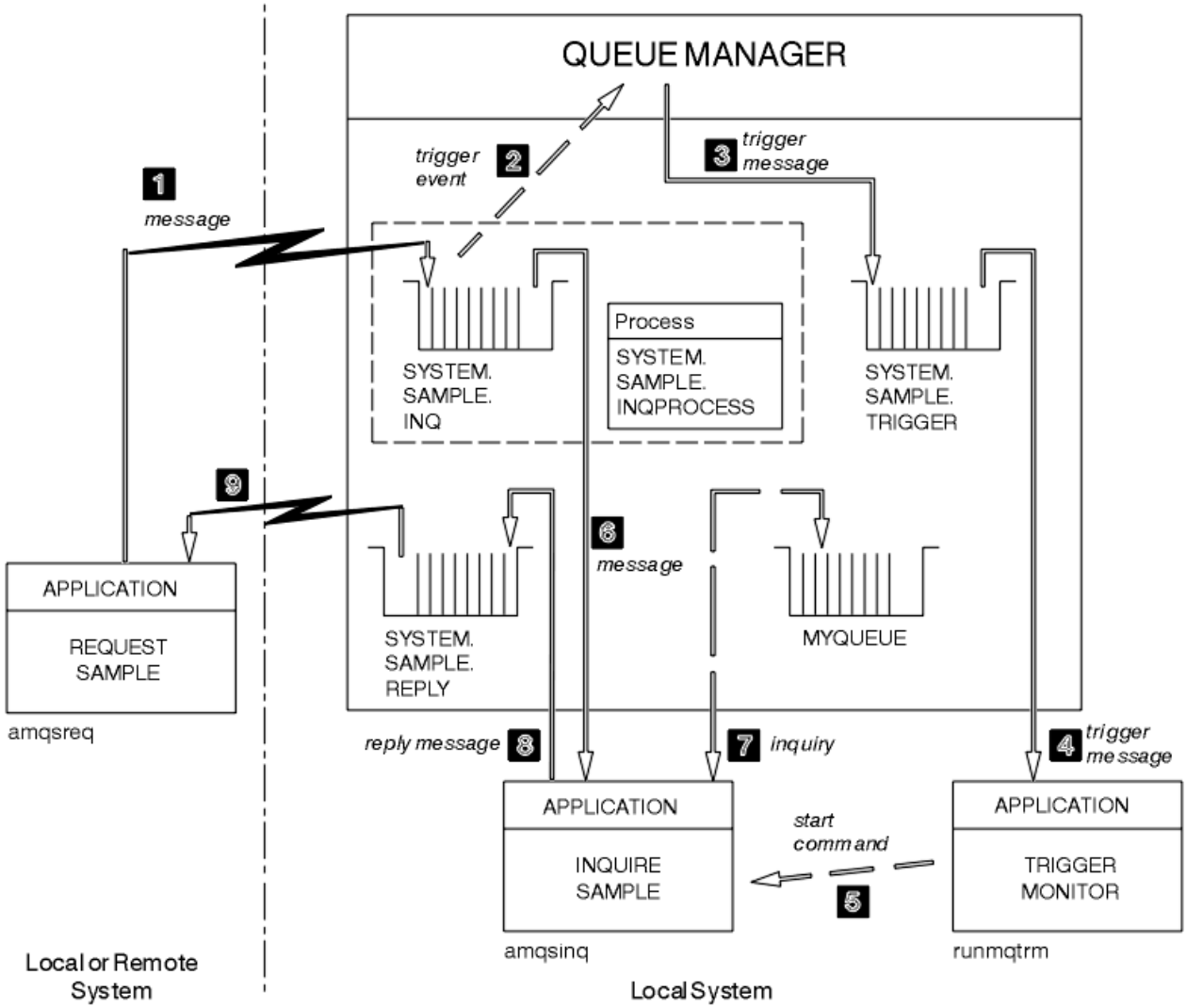
   Tyto fronty mají typ spouštěče FIRST, takže pokud již ve frontě existují zprávy před spuštěním ukázky Požadavek, serverové aplikace se nespustí pomocí zpráv, které odešlete.

4. Ujistěte se, že jste definovali frontu pro ukázkový program Inquire, Set nebo Echo, který se má použít.

To znamená, že je monitor spouštěčů připraven, když ukázka požadavku odešle zprávu.

Poznámka: Ukázkové definice procesu vytvořené pomocí RUNMQSC a amqscos0.tst spustí ukázky jazyka C. Změňte definice procesu v souboru amqscos0.tst a použijte RUNMQSC s tímto aktualizovaným souborem pro použití verzí jazyka COBOL.

Obrázek 22 na stránce 143 demonstruje, jak používat ukázky Požadavek a Dotázat se dohromady.



Obrázek 22. Požadavek a zjišťování ukázek pomocí spouštěče

Do pole Obrázek 22 na stránce 143 Vzorek požadavku vloží zprávy do cílové fronty serveru SYSTEM.SAMPLE.INQa Dotaz na ukázku se dotazuje fronty, MYQUEUE. Případně můžete použít jednu z ukázkových front definovaných při spuštění příkazu amqscos0.tstnebo jakékoli jiné fronty, kterou jste definovali, pro ukázku Inquire.

Poznámka: Čísla v produktu Obrázek 22 na stránce 143 zobrazují posloupnost událostí.

Chcete-li spustit ukázky Požadavek a odběr, použijte spuštění:

1. Zkontrolujte, zda jsou definované fronty, které chcete použít, definovány. Spusťte amqscos0.tst, abyste definovali ukázkové fronty, a definujte frontu MYQUEUE.
2. Spusťte příkaz RUNMQTRM monitoru spouštěčů:

```
RUNMQTRM -m qmanagername -q SYSTEM.SAMPLE.TRIGGER
```

3. Spustit ukázku požadavku

```
amqsreq SYSTEM.SAMPLE.INQ
```

   Poznámka: Objekt procesu definuje, co se má spustit. Pokud klient a server nejsou spuštění na stejné platformě, musí všechny procesy spuštěné monitorem spouštěčů definovat *ApplType*, jinak server převezme své výchozí definice (tj. typ aplikace, která je obvykle přidružena k počítači serveru) a způsobí selhání.

   Seznam typů aplikací viz ApplType.

4. Zadejte název fronty, kterou chcete použít jako vzorek pro zjišťování:

```
MYQUEUE
```

5. Zadejte prázdný řádek (chcete-li ukončit Požadavek na program).
6. Ukázka požadavku pak zobrazí zprávu obsahující data, která program Inquire získal ze MYQUEUE.

Můžete použít více než jednu frontu; v tomto případě zadejte názvy ostatních front v kroku “4” na stránce 144.

Další informace o spouštění viz “Spuštění aplikací produktu IBM WebSphere MQ pomocí spouštěčů” na stránce 316.

Návrh ukázkového programu Požadavek

Program otevře cílovou frontu serveru tak, aby mohla vkládat zprávy. Používá volání MQOPEN s volbou MQOO_OUTPUT. Nemůže-li frontu otevřít, zobrazí se v programu chybová zpráva obsahující kód příčiny vrácený voláním MQOPEN.

Program pak otevře frontu pro odpověď s názvem SYSTEM.SAMPLE.REPLY , aby bylo možné získat zprávy odpovědí. Za tímto způsobem program používá volání MQOPEN s volbou MQOO_INPUT_EXCLUSIVE. Nemůže-li frontu otevřít, zobrazí se v programu chybová zpráva obsahující kód příčiny vrácený voláním MQOPEN.

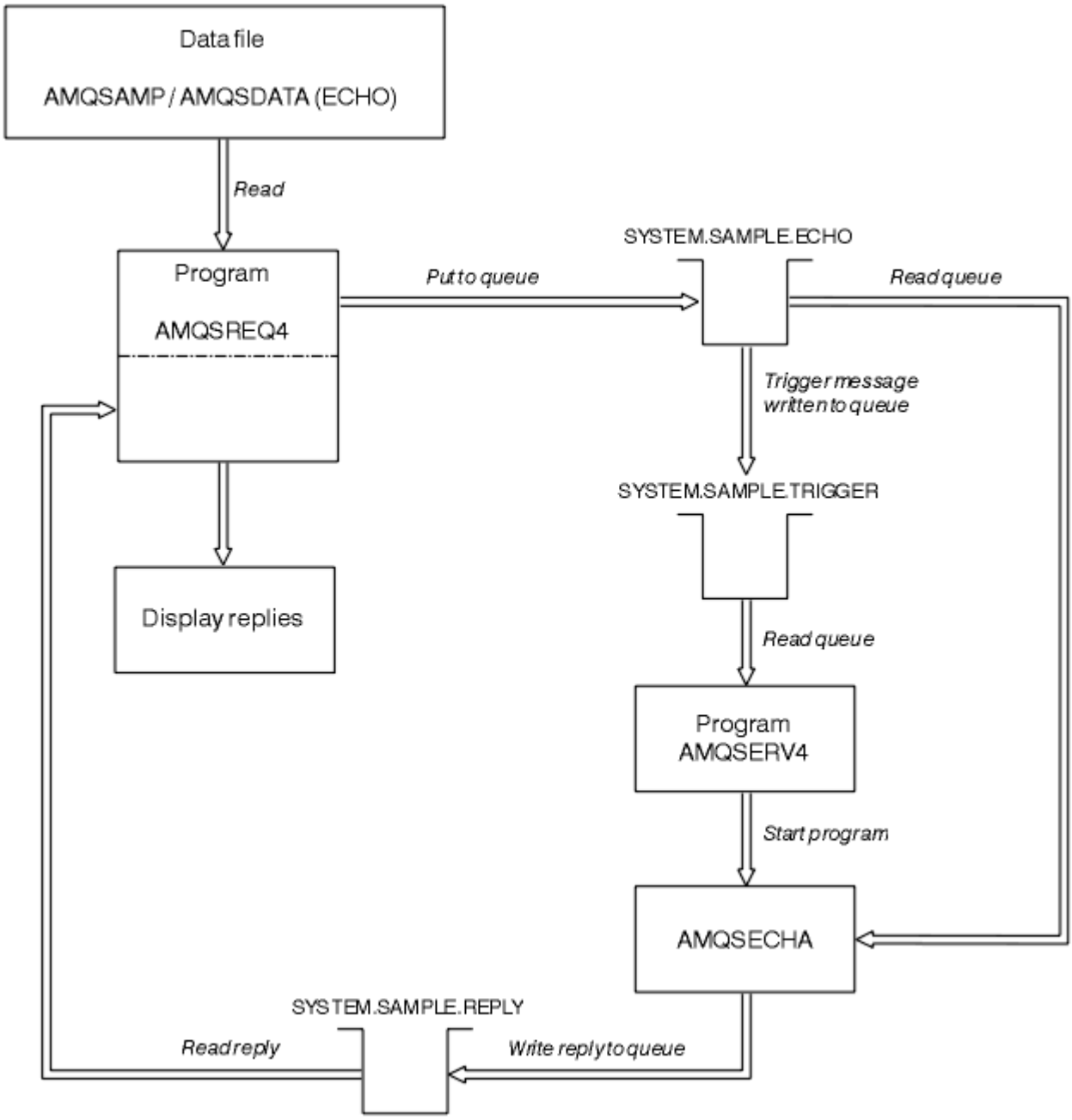
Pro každý řádek vstupu program pak přečte text do vyrovnávací paměti a použije volání MQPUT k vytvoření zprávy požadavku obsahující text této řádky. Při tomto volání program používá volbu sestavy MQRO_EXCEPTION_WITH_DATA, aby bylo možné požádat, aby všechny zprávy sestavy odeslané o zprávě požadavku obsahovaly prvních 100 bajtů dat zprávy. Program pokračuje, dokud nedojde k dosažení konce vstupu nebo volání MQPUT selže.

Program pak použije volání MQGET k odstranění zpráv odpovědí z fronty a zobrazí data obsažená v odpovědích. Volání MQGET používá volby MQGMO_WAIT, MQGMO_CONVERT a MQGMO_ACCEPT_TRUNCATED. *WaitInterval* je 5 minut ve verzi jazyka COBOL a 1 minutu ve verzi C, pro první odpověď (k povolení času pro aplikaci serveru) a 15 sekund pro následné odpovědi. Program čeká na tato období, pokud ve frontě není žádná zpráva. Pokud před uplynutím tohoto intervalu nepřijde žádná zpráva, volání se nezdaří a vrátí kód příčiny MQRC_NO_MSG_AVAILABLE. Volání také používá volbu MQGMO_ACCEPT_TRUNCATED_MSG, takže zprávy, které jsou delší než deklarovaná velikost vyrovnávací paměti, jsou oříznuty.

Tento program ukazuje, jak vymazat pole *MsgId* a *CorrelId* struktury MQMD po každém volání MQGET, protože volání nastavuje tato pole na hodnoty obsažené ve zprávě, kterou načítá. Vymazání těchto polí znamená, že po sobě jdoucí příkazy MQGET načítají zprávy v pořadí, ve kterém jsou zprávy zadrženy ve frontě.

Program pokračuje, dokud buď volání MQGET nevrátí kód příčiny MQRC_NO_MSG_AVAILABLE, nebo se volání MQGET nezdaří. Pokud se volání nezdaří, zobrazí program chybovou zprávu, která obsahuje kód příčiny.

Program pak zavře jak frontu cílového serveru, tak frontu pro odpověď pomocí volání MQCLOSE.



Obrázek 23. Ukázkový graf vývojového diagramu IBM i Klient/Server (Echo)

Ukázkové programy Set

Ukázkové programy brání vložení operací do fronty pomocí volání MQSET za účelem změny atributu *InhibitPut* fronty. Také se dozvíte o návrhu ukázkových programů.

Názvy těchto programů viz “Vlastnosti demonstrovány v ukázkových programech” na stránce 94 .

Programy jsou určeny ke spuštění jako spuštěné programy, takže jejich jediným vstupem je struktura MQTMC2 (zpráva spouštěče), která obsahuje název cílové fronty s atributy, které mají být dotazovány. Verze jazyka C také používá název správce front. Verze COBOL používá výchozího správce front.

Aby spouštěcí proces fungoval, ujistěte se, že ukázkový program Nastavení, který chcete použít, je spuštěn zprávami přicházejícími do fronty SYSTEM.SAMPLE.SET. Chcete-li to provést, zadejte název ukázkového programu Set, který chcete použít v poli *ApplicId* definice procesu SYSTEM.SAMPLE.SETPROCESS. Ukázková fronta má typ spouštěče FIRST; pokud již ve frontě existují zprávy, než spustíte ukázku Požadavek, nespustí se ukázka Nastavit zprávy, které jste odeslali.

Pokud jste správně nastavili definici:

- Pro systémy UNIX, Linux a Windows spusťte program **runmqtrm** v jedné relaci a pak spusťte program amqsreq v jiném.
- V případě produktu IBM ispusťte program AMQSERV4 v jedné relaci a poté spusťte program AMQSREQ4 v jiném. Můžete použít AMQSTRG4 místo AMQSERV4, ale potenciální zpoždění při odeslání úlohy by mohla méně snadno sledovat, co se děje.

Ukázkové programy požadavku slouží k odeslání zpráv požadavků, z nichž každá obsahuje pouze název fronty, do fronty SYSTEM.SAMPLE.SET. Pro každou zprávu požadavku odešlete ukázkové programy odeslání zprávy odpovědi obsahující potvrzení, že operace put byly na zadané frontě zablokovány. Odpovědi se posílají do fronty odpovědí uvedené ve zprávě s požadavkem.

Návrh ukázkového programu Set

Program otevře frontu uvedenou ve struktuře zpráv spouštěče, která byla předána, když byla spuštěna. (Pro srozumitelnost zavoláme tuto *frontu požadavků*.) Program používá volání MQOPEN k otevření této fronty pro sdílený vstup.

Program používá volání MQGET k odstranění zpráv z této fronty. Toto volání používá volby MQGMO_ACCEPT_TRUNCATED_MSG a MQGMO_WAIT, s intervalem čekání 5 sekund. Program testuje deskriptor každé zprávy za účelem zjištění, zda se jedná o zprávu požadavku; pokud není, program zahodí zprávu a zobrazí varovnou zprávu.

Pro každou zprávu požadavku odebranou z fronty požadavků tento program přečte název fronty (kterou nazýváme *cílovou frontou*) obsaženou v datech a otevře tuto frontu pomocí volání MQOPEN s volbou MQOO_SET. Program potom použije volání MQSET k nastavení hodnoty atributu *InhibitPut* cílové fronty na hodnotu MQQA_PUT_INHIBITED.

Je-li volání MQSET úspěšné, program použije volání MQPUT1 k vložení zprávy odpovědi do fronty pro odpovědi. Tato zpráva obsahuje řetězec `PUT inhibited`.

Je-li volání MQOPEN nebo MQSET neúspěšné, program použije volání MQPUT1 k vložení zprávy produktu `report` do fronty pro odpovědi. V poli *Feedback* v deskriptoru zprávy této zprávy je kód příčiny vrácený voláním MQOPEN nebo MQSET, v závislosti na tom, který z nich selhal.

Po volání MQSET program zavře cílovou frontu pomocí volání MQCLOSE.

Pokud ve frontě požadavků nejsou žádné zprávy, program tuto frontu zavře a odpojí se od správce front.

Ukázkový program SSL/TLS

AMQSSLC je ukázkový program C, který ukazuje, jak používat struktury MQCNO a MQSCO k poskytnutí informací o připojení klienta SSL/TLS v rámci volání MQCONNX. To umožňuje aplikaci klienta MQI poskytnout definici svého kanálu připojení klienta a nastavení SSL/TLS v běhovém prostředí bez tabulky CCDT (Client Channel Definition table).

Je-li zadán název připojení, vytvoří program definici kanálu připojení klienta ve struktuře MQCD.

Je-li zadán název kmene souboru úložiště klíčů, vytvoří program strukturu MQSCO; pokud je také dodána adresa URL odpovídacího modulu OCSP, program vytvoří strukturu MQAIR záznamu ověřovacích informací.

Program se poté připojí ke správci front pomocí příkazu MQCONNX. Rozvodí a vytiskne název správce front, ke kterému je připojen.

Tento program je určen k propojení s aplikací klienta MQI. Lze ji však propojit jako běžnou aplikaci MQI. Pak se jednoduše připojí k lokálnímu správci front a bude ignorovat informace o připojení klienta.

AMQSSLC přijímá následující parametry, všechny jsou volitelné:

-m QmgrName
Název správce front, ke kterému se má připojit

-c ChannelName
Název kanálu, který má být použit

-x ConnName
Název připojení serveru

Parametry SSL/TLS

-k KeyReposStem
Název kmene souboru úložiště klíčů. Jedná se o úplnou cestu k souboru bez přípony .kdb. Příklad:

```
/home/user/client
C:\User\client
```

-s CipherSpec
Řetězec CipherSpec kanálu SSL/TLS odpovídající hodnotě SSLCIPH v definici kanálu SVRCONN ve správci front.

-f
Uvádí, že musí být použity pouze certifikované algoritmy FIPS 140-2.

-b VALUE1[,VALUE2...]
Určuje, že musí být použity pouze algoritmy standardu Suite B. Tento parametr je seznam s čárkami jako oddělovači jedné nebo více následujících hodnot: NONE,128_BIT,192_BIT. Tyto hodnoty mají stejný význam jako hodnoty proměnné prostředí MQSUITEB a ekvivalentní nastavení EncryptionPolicySuiteB v sekci konfiguračního souboru klienta zabezpečení SSL.

-p Zásada
Uvádí zásadu ověření certifikátu, která se má použít. Může jít o jednu z následujících hodnot:

ANY
Použít všechny zásady ověření platnosti certifikátů podporované knihovnou zabezpečených soketů a přijmout řetěz certifikátů, pokud některý ze zásad považuje řetěz certifikátů za platný. Toto nastavení lze použít pro maximální zpětnou kompatibilitu se staršími digitálními certifikáty, které nesplňují moderní certifikační standardy.

RFC5280
Použít pouze zásadu ověření platnosti certifikátu vyhovujícího RFC 5280. Toto nastavení poskytuje přísnější validaci než nastavení ANY, ale odmítá některé starší digitální certifikáty.

Výchozí hodnota je ANY.

Parametr odvolání certifikátu OCSP:

-o adresa URL
Adresa URL odpovídacího modulu OCSP

Spuštění ukázkového programu SSL/TLS

Chcete-li spustit ukázkový program SSL/TLS, musíte nejprve nastavit prostředí SSL nebo TLS. Poté spustíte ukázku z příkazového řádku a dodáte tak počet parametrů.

Informace o této úloze

Následující pokyny spouštějí ukázkový program s použitím osobních certifikátů. Tímto příkazem můžete například pomocí certifikátu CA používat certifikáty certifikačních autorit a kontrolovat jejich stav pomocí odpovídacího modulu OCSP. Viz pokyny v rámci ukázky.

Postup

1. Vytvořte správce front s názvem QM1. Další informace viz crtmqm.
2. Vytvořte úložiště klíčů pro správce front. Další informace naleznete v tématu Nastavení úložiště klíčů v systémech UNIX, Linux, and Windows.
3. Vytvořte úložiště klíčů pro klienta. Nazvěte jej *clientkey.kdb*.

4. Vytvořte osobní certifikát pro správce front. Další informace viz téma Vytvoření osobního certifikátu s automatickým podpisem na systémech UNIX, Linux, and Windows.
5. Vytvořte osobní certifikát pro klienta.
6. Extrahujte osobní certifikát z úložiště klíčů serveru a přidejte jej do úložiště klienta. Další informace naleznete v tématu Extrakce veřejné části certifikátu podepsaného držitelem z úložiště klíčů v systémech UNIX, Linux a Windows a Přidání certifikátu CA (nebo veřejné části certifikátu podepsaného držitelem) do úložiště klíčů v systémech UNIX, Linux nebo Windows.
7. Extrahujte osobní certifikát z úložiště klíčů klienta a přidejte jej do úložiště klíčů serveru.
8. Vytvořte kanál připojení k serveru pomocí příkazu MQSC:

   ```
   DEFINE CHANNEL(QM1SVRCONN) CHLTYPE(SVRCONN) TRPTYPE(TCP) SSLCIPH(NULL_SHA)
   ```

   Další informace naleznete v tématu Kanál připojení serveru .
9. Definujte a spusťte modul listener kanálu na správci front. Další informace viz DEFINE LISTENER a START LISTENER.
10. Spusťte ukázkový program pomocí následujícího příkazu:

    ```
    AMQSSSLC -m QM1 -c QM1SVRCONN -x localhost
    -k "c:\Program Files\IBM\WebSphere MQ\clientkey" -s NULL_SHA
    -o http://dummy.OCSP.responder
    ```

Výsledky

Ukázkový program provádí následující akce:

1. Připojí se k libovolnému určenému správci front nebo k výchozímu správci front s použitím zadaných voleb.
2. Otevře správce front a ve svém názvu zklidní.
3. Zavře správce front.
4. Odpojí se od správce front.

Pokud se ukázkový program spustí úspěšně, zobrazí výstup podobný následujícímu příkladu:

```
Sample AMQSSSLC start
Connecting to queue manager QM1
Using the server connection channel QM1SVRCONN
on connection name localhost.
Using SSL CipherSpec NULL_SHA
Using SSL key repository stem c:\Program Files\IBM\WebSphere MQ\clientkey
Using OCSP responder URL http://dummy.OCSP.responder
Connection established to queue manager QM1

Sample AMQSSSLC end
```

Pokud ukázkový program narazí na problém, zobrazí příslušnou chybovou zprávu, například pokud zadáte neplatnou adresu URL odpovídacího modulu OCSP, obdržíte následující zprávu:

```
MQCONNX ended with reason code 2553
```

Seznam kódů příčiny najdete v tématu Kódy příčiny rozhraní API.

Spouštěcí ukázkové programy

Funkce poskytnutá ve spouštěcí ukázce je podmnožinou, která je poskytována v monitoru spouštěčů v programu **runmqtrm** .

Názvy těchto programů viz “Vlastnosti demonstrovány v ukázkových programech” na stránce 94 .

Návrh spouštěcí ukázky

Spouštěcí ukázkový program otevře inicializační frontu s použitím volání MQOPEN s volbou MQOO_INPUT_AS_Q_DEF. Zpráva získává zprávy z inicializační fronty pomocí příkazu MQGET s volbami MQGMO_ACCEPT_TRUNCATED_MSG a MQGMO_WAIT, přičemž určuje neomezený interval čekání. Program vymaže pole *MsgId* a *CorrelId* před každým voláním MQGET, aby získal zprávy v posloupnosti.

Po načtení zprávy z inicializační fronty program otestuje zprávu tak, že zkontroluje velikost zprávy a ujistěte se, že má stejnou velikost jako struktura MQTM. Pokud tento test selže, zobrazí se v programu varování.

V případě platných zpráv spouštěče spouštěcí ukázka kopíruje data z těchto polí: *ApplicId*, *EnvrData*, *Versio*n a *ApplType*. Poslední dvě z těchto polí jsou numerická, takže program vytvoří náhradu znaků pro použití ve struktuře MQTMC2 pro systémy UNIX, Linuxa Windows .

Ukázka spouštěče vydá spouštěcí příkaz do aplikace zadané v poli *ApplicId* zprávy spouštěče a předává strukturu MQTMC2 nebo MQTMC (znaková verze zprávy spouštěcího impulsu). V systémech UNIX, Linux a Windows se pole *EnvData* používá jako rozšíření pro řetězec příkazu vyvolání.

Nakonec program zavře inicializační frontu.

Spuštění ukázkových programů spouštění

Toto téma obsahuje informace o spouštění ukázkových programů spouštění.

Spuštění ukázek amqstrg0.c, amqstrg, a amqstrgc

Program má 2 parametry:

1. Název inicializační fronty (nezbytné)
2. Název správce front (volitelný)

Není-li správce front zadán, připojí se k výchozímu správci front. Ukázková inicializační fronta bude definována při spuštění příkazu amqscos0.tst; název této fronty: SYSTEM.SAMPLE.TRIGGERa vy jej můžete použít při spuštění tohoto programu.

Poznámka: Funkce v této ukázce je podmnožinou úplné spouštěcí funkce, která je dodána v programu **runmqtrm** .

Návrh spouštěcího serveru

Návrh spouštěcího serveru je podobný jako u monitoru spouštěčů, kromě toho, že spouštěcí server:

- Umožňuje použití aplikací MQAT_CICS a aplikací MQAT_OS400 .
- Pro aplikace CICS substituuje produkt *EnvData*, například, chcete-li určit oblast CICS , ze zprávy spouštěče v příkazu STRCICSUSR
- Otevře inicializační frontu pro sdílený vstup, takže mnoho serverů spouštěčů může být spuštěno ve stejnou dobu.

Poznámka: Programy, které spustil AMQSERV4 , nesmí používat volání MQDISC, protože tento server zastaví spouštěcí server. Pokud programy spouštěné systémem AMQSERV4 používají volání MQCONN, získají kód příčiny MQRC_ALREADY_CONNECTED.

Ukázky TUXEDO

Zde najdete informace o ukázkových programech pro operaci Put and Get pro prostředí TUXEDO a pro sestavení prostředí serveru v TUXEDO.

Před spuštěním těchto ukázek je třeba vytvořit prostředí serveru.

Poznámka: V celém tomto tématu se znak zpětného lomítka (\) používá k rozdělení dlouhých příkazů na více než jeden řádek. Nezadávejte tento znak. Zadejte každý příkaz jako jeden řádek.

Vytváření prostředí serveru

Informace o vytváření prostředí serveru pro produkt WebSphere MQ pro různé platformy.

Předpokládá se, že máte fungující prostředí TUXEDO.

Sestavování prostředí serveru pro produkt WebSphere MQ for AIX (32bitový)

1. Vytvořte adresář (například < APPDIR>), v němž je založeno prostředí serveru, a proveďte všechny příkazy v tomto adresáři.
2. Exportujte následující proměnné prostředí, kde TUXDIR je kořenový adresář pro TUXEDO, a *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ instalován:

```
$ export CFLAGS="-I MQ_INSTALLATION_PATH□inc -I /<APPDIR> -L MQ_INSTALLATION_PATH□lib"
$ export LDOPTS="-lmqm"
$ export FIELDTBLS=MQ_INSTALLATION_PATH/samp/amqstxvx.flds
$ export VIEWFILES=/<APPDIR>/amqstxvx.V
$ export LIBPATH=$TUXDIR□lib:MQ_INSTALLATION_PATH□lib:□lib
```

3. Přidejte do souboru udataobj/RM souboru TUXEDO následující:

```
MQSeries_XA_RMI:MQRMIXASwitchDynamic: -lmqmxa -lmqm
```

4. Spusťte příkazy:

```
$ mkfldhdr    MQ_INSTALLATION_PATH/samp/amqstxvx.flds
$ viewc       MQ_INSTALLATION_PATH/samp/amqstxvx.v
$ buildtms    -o MQXA -r MQSeries_XA_RMI
$ buildserver -o MQSERV1 -f MQ_INSTALLATION_PATH/samp/amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.a \
       -r MQSeries_XA_RMI -s MPUT1:MPUT \
       -s MGET1:MGET \
       -v -bshm
$ buildserver -o MQSERV2 -f MQ_INSTALLATION_PATH/samp/amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.a \
       -r MQSeries_XA_RMI -s MPUT2:MPUT
       -s MGET2:MGET \
       -v -bshm
$ buildclient -o doputs  -f MQ_INSTALLATION_PATH/samp/amqstxpx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.a
$ buildclient -o dogets  -f MQ_INSTALLATION_PATH/samp/amqstxgx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.a
```

5. Upravte soubor ubbstxcx.cfg a v případě potřeby přidejte podrobnosti o názvu počítače, pracovních adresářích a správci front:

```
$ tmloadcf    -y MQ_INSTALLATION_PATH/samp/ubbstxcx.cfg
```

6. Vytvořte TLOGDEVICE:

```
$tmadmin -c
```

   Zobrazí se výzva k zadání. Na této obrazovce zadejte:

```
> crdl -z /<APPDIR>/TLOG1
```

7. Spusťte správce front:

```
$ strmqm
```

8. Spustit Tuxedo

```
$ tmboot -y
```

Nyní můžete pomocí programů doputs a dogets vkládat zprávy do fronty a načítat je z fronty.

Sestavení prostředí serveru pro produkt WebSphere MQ for AIX (64bitová verze)

1. Vytvořte adresář (například < APPDIR>), v němž je založeno prostředí serveru, a proveďte všechny příkazy v tomto adresáři.
2. Exportujte následující proměnné prostředí, kde TUXDIR představuje kořenový adresář pro TUXEDO, a *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ installed.:

```
$ export CFLAGS="-I MQ_INSTALLATION_PATH□inc -I /<APPDIR> -L
MQ_INSTALLATION_PATH□lib64"
$ export LDOPTS="-lmqm"
$ export FIELDTBLS=MQ_INSTALLATION_PATH/samp/amqstxvx.flds
$ export VIEWFILES=/<APPDIR>/amqstxvx.V
$ export LIBPATH=$TUXDIR□lib64:MQ_INSTALLATION_PATH□lib64:□lib64
```

3. Přidejte do souboru udataobj/RM souboru TUXEDO následující:

```
MQSeries_XA_RMI:MQRMIXASwitchDynamic: -lmqmxa64 -lmqm
```

4. Spusťte příkazy:

```
$ mkfldhdr    MQ_INSTALLATION_PATH/samp/amqstxvx.flds
$ viewc       MQ_INSTALLATION_PATH/samp/amqstxvx.v
$ buildtms    -o MQXA -r MQSeries_XA_RMI
$ buildserver -o MQSERV1 -f MQ_INSTALLATION_PATH/samp/amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.a \
       -r MQSeries_XA_RMI -s MPUT1:MPUT \
       -s MGET1:MGET \
       -v -bshm
$ buildserver -o MQSERV2 -f MQ_INSTALLATION_PATH/samp/amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.a \
       -r MQSeries_XA_RMI -s MPUT2:MPUT
       -s MGET2:MGET \
       -v -bshm
$ buildclient -o doputs  -f MQ_INSTALLATION_PATH/samp/amqstxpx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.a
$ buildclient -o dogets  -f MQ_INSTALLATION_PATH/samp/amqstxgx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.a
```

5. Upravte soubor ubbstxcx.cfg a v případě potřeby přidejte podrobnosti o názvu počítače, pracovních adresářích a správci front:

```
$ tmloadcf    -y MQ_INSTALLATION_PATH/samp/ubbstxcx.cfg
```

6. Vytvořte TLOGDEVICE:

```
$tmadmin -c
```

   Zobrazí se výzva k zadání. Na této obrazovce zadejte:

```
> crdl -z /<APPDIR>/TLOG1
```

7. Spusťte správce front:

```
$ strmqm
```

8. Spustit Tuxedo

```
$ tmboot -y
```

Nyní můžete pomocí programů doputs a dogets vkládat zprávy do fronty a načítat je z fronty.

Sestavení prostředí serveru pro produkt WebSphere MQ for Solaris (32bitový)

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

1. Vytvořte adresář (například *APPDIR*), v němž je prostředí serveru sestaveno a proveďte všechny příkazy v tomto adresáři.
2. Vyexportujte následující proměnné prostředí, kde TUXDIR je kořenový adresář pro TUXEDO:

```
$ export CFLAGS="-I /APPDIR"
$ export FIELDTBLS=amqstxvx.flds
$ export VIEWFILES=amqstxvx.V
$ export SHLIB_PATH=$TUXDIR/lib:MQ_INSTALLATION_PATH/lib:/lib
$ export LD_LIBRARY_PATH=$TUXDIR/lib:MQ_INSTALLATION_PATH/lib:/lib
```

3. Přidejte následující do souboru udataobj/RM TUXEDO.

```
MQSeries_XA_RMI:MQRMIXASwitchDynamic: \
MQ_INSTALLATION_PATH/lib/libmqmxa.a MQ_INSTALLATION_PATH/lib/libmqm.so \
/opt/tuxedo/lib/libtux.a
```

4. Spusťte příkazy:

```
$ mkfldhdr    amqstxvx.flds
$ viewc       amqstxvx.v
$ buildtms    -o MQXA -r MQSeries_XA_RMI
$ buildserver -o MQSERV1 -f amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.so \
       -r MQSeries_XA_RMI -s MPUT1:MPUT \
       -s MGET1:MGET \
       -v -bshm
       -l -ldl
$ buildserver -o MQSERV2 -f amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.so \
       -r MQSeries_XA_RMI -s MPUT2:MPUT \
       -s MGET2:MGET \
       -v -bshm
       -l -ldl
$ buildclient -o doputs  -f amqstxpx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.so  \

$ buildclient -o dogets  -f amqstxgx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.so
```

5. Upravte soubor ubbstxcx.cfg a v případě potřeby přidejte podrobnosti o názvu počítače, pracovních adresářích a správci front:

```
$ tmloadcf    -y ubbstxcx.cfg
```

6. Vytvořte TLOGDEVICE:

```
$tmadmin -c
```

Zobrazí se výzva k zadání. Na této obrazovce zadejte:

```
> crdl -z /APPDIR/TLOG1
```

7. Spusťte správce front:

```
$ strmqm
```

8. Spustit Tuxedo

```
$ tmboot -y
```

Nyní můžete pomocí programů doputs a dogets vkládat zprávy do fronty a načítat je z fronty.

Sestavení prostředí serveru pro produkt WebSphere MQ for Solaris (64bitový)

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

1. Vytvořte adresář (například < APPDIR>), v němž je založeno prostředí serveru, a proveďte všechny příkazy v tomto adresáři.
2. Vyexportujte následující proměnné prostředí, kde TUXDIR je kořenový adresář pro TUXEDO:

```
$ export CFLAGS="-I /<APPDIR>"
$ export FIELDTBLS=amqstxvx.flds
$ export VIEWFILES=amqstxvx.V
$ export SHLIB_PATH=$TUXDIR□lib:MQ_INSTALLATION_PATH□lib:□lib64
$ export LD_LIBRARY_PATH=$TUXDIR□lib64:MQ_INSTALLATION_PATH□lib64:□lib64
```

3. Přidejte následující do souboru udataobj/RM TUXEDO.

```
MQSeries_XA_RMI:MQRMIXASwitchDynamic: \
MQ_INSTALLATION_PATH/lib64/libmqmxa64.a MQ_INSTALLATION_PATH/lib64/libmqm.so \
/opt/tuxedo/lib64/libtux.a
```

4. Spusťte příkazy:

```
$ mkfldhdr    amqstxvx.flds
$ viewc       amqstxvx.v
$ buildtms    -o MQXA -r MQSeries_XA_RMI
$ buildserver -o MQSERV1 -f amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.so \
       -r MQSeries_XA_RMI -s MPUT1:MPUT \
       -s MGET1:MGET \
       -v -bshm
       -l -ldl
$ buildserver -o MQSERV2 -f amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.so \
       -r MQSeries_XA_RMI -s MPUT2:MPUT \
       -s MGET2:MGET \
       -v -bshm
       -l -ldl
$ buildclient -o doputs  -f amqstxpx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.so  \

$ buildclient -o dogets  -f amqstxgx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.so
```

5. Upravte soubor ubbstxcx.cfg a v případě potřeby přidejte podrobnosti o názvu počítače, pracovních adresářích a správci front:

```
$ tmloadcf    -y ubbstxcx.cfg
```

6. Vytvořte TLOGDEVICE:

```
$tmadmin -c
```

Zobrazí se výzva k zadání. Na této obrazovce zadejte:

```
> crdl -z /<APPDIR>/TLOG1
```

7. Spusťte správce front:

```
$ strmqm
```

8. Spustit Tuxedo

```
$ tmboot -y
```

Nyní můžete pomocí programů doputs a dogets vkládat zprávy do fronty a načítat je z fronty.

Sestavení prostředí serveru pro produkt WebSphere MQ for HP-UX (32bitový)

Poznámka: 32bitové prostředí serveru TUXEDO může být vytvořeno pouze na platformě Itanium.

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

1. Vytvořte adresář (například < APPDIR>), v němž je založeno prostředí serveru, a proveďte všechny příkazy v tomto adresáři.
2. Vyexportujte následující proměnné prostředí, kde TUXDIR je kořenový adresář pro TUXEDO:

```
$ export CFLAGS="-Aa -D_HPUX_SOURCE"
$ export FIELDTBLS=MQ_INSTALLATION_PATH/samp/amqstxvx.flds
$ export VIEWFILES=$APPDIR/amqstxvx.V
$ export TUXCONFIG=$APPDIR/tuxconfig
$ export PATH=$TUXDIR/bin:/usr/bin:/sbin:MQ_INSTALLATION_PATH/bin:$PATH
$ export SHLIB_PATH=$TUXDIR/lib:MQ_INSTALLATION_PATH/lib:/lib
$ export FLDTBLDIR=$APPDIR:$TUXDIR/udataobj
```

3. Přidejte do souboru udataobj/RM souboru TUXEDO následující:

```
MQSeries_XA_RMI:MQRMIXASwitchDynamic: \
MQ_INSTALLATION_PATH/lib/libmqmxa.so MQ_INSTALLATION_PATH/lib/libmqm.so \
/opt/tuxedo/lib/libtux.sl
```

4. Spusťte příkazy:

```
$ mkfldhdr    MQ_INSTALLATION_PATH/samp/amqstxvx.flds
$ viewc       MQ_INSTALLATION_PATH/samp/amqstxvx.v
```

Po spuštění příkazů mkfldhdr a viewc se vytvoří soubor záhlaví amqstxvx.h v adresáři aplikace TUXEDO. Zkopírujte tento soubor z adresáře aplikace TUXEDO do adresáře include TUXEDO a pak spusťte následující příkazy.

```
$ buildtms    -o MQXA -r MQSeries_XA_RMI
$ buildserver -o MQSERV1 -f MQ_INSTALLATION_PATH/samp/amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.so \
       -r MQSeries_XA_RMI -s MPUT1:MPUT \
       -s MGET1:MGET \
       -v -bshm
$ buildserver -o MQSERV2 -f MQ_INSTALLATION_PATH/samp/amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.so \
       -r MQSeries_XA_RMI -s MPUT2:MPUT \
       -s MGET2:MGET \
       -v -bshm
$ buildclient -o doputs  -f MQ_INSTALLATION_PATH/samp/amqstxpx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.so
$ buildclient -o dogets  -f MQ_INSTALLATION_PATH/samp/amqstxgx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.so
```

5. Upravte soubor ubbstxcx.cfg a v případě potřeby přidejte podrobnosti o názvu počítače, pracovních adresářích a správci front:

```
$ tmloadcf    -y MQ_INSTALLATION_PATH/samp/ubbstxcx.cfg
```

6. Vytvořte TLOGDEVICE:

```
$tmadmin -c
```

Zobrazí se výzva k zadání. Na této obrazovce zadejte:

```
> crdl -z /<APPDIR>/TLOG1
```

7. Spusťte správce front:

```
$ strmqm
```

8. Spustit TUXEDO:

```
$ tmboot -y
```

Nyní můžete pomocí programů doputs a dogets vkládat zprávy do fronty a načítat je z fronty.

Sestavení prostředí serveru pro produkt WebSphere MQ for HP-UX (64bitová verze)

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

1. Vytvořte adresář (například < APPDIR>), v němž je založeno prostředí serveru, a proveďte všechny příkazy v tomto adresáři.
2. Vyexportujte následující proměnné prostředí, kde TUXDIR je kořenový adresář pro TUXEDO:

```
$ export CFLAGS="-Aa -D_HPUX_SOURCE"
$ export FIELDTBLS=MQ_INSTALLATION_PATH/samp/amqstxvx.flds
$ export VIEWFILES=$APPDIR/amqstxvx.V
$ export TUXCONFIG=$APPDIR/tuxconfig
$ export PATH=$TUXDIR/bin:/usr/bin:/sbin:MQ_INSTALLATION_PATH/bin:$PATH
$ export SHLIB_PATH=$TUXDIR/lib:MQ_INSTALLATION_PATH/lib64:/lib64
$ export FLDTBLDIR=$APPDIR:$TUXDIR/udataobj
```

3. Přidejte do souboru udataobj/RM souboru TUXEDO následující:

Na platformě HP-UX IA64 (IPF):

```
MQSeries_XA_RMI:MQRMIXASwitchDynamic: \
MQ_INSTALLATION_PATH/lib64/libmqmxa64.so MQ_INSTALLATION_PATH/lib64/libmqm.so \
/opt/tuxedo/lib/libtux.sl
```

Poznámka: Knihovny produktu WebSphere MQ dodávané na platformě HP-UX IA64 (IPF) mají příponu názvu souboru .so.

4. Spusťte příkazy:

```
$ mkfldhdr    MQ_INSTALLATION_PATH/samp/amqstxvx.flds
$ viewc       MQ_INSTALLATION_PATH/samp/amqstxvx.v
```

Po spuštění příkazů mkfldhdr a viewc se vytvoří soubor záhlaví amqstxvx.h v adresáři aplikace TUXEDO. Zkopírujte tento soubor z adresáře aplikace TUXEDO do adresáře include TUXEDO a pak spusťte následující příkazy.

```
$ buildtms    -o MQXA -r MQSeries_XA_RMI
```

Na platformě HP-UX IA64 (IPF):

```
$ buildserver -o MQSERV1 -f MQ_INSTALLATION_PATH/samp/amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.so \
       -r MQSeries_XA_RMI -s MPUT1:MPUT \
       -s MGET1:MGET \
       -v -bshm
$ buildserver -o MQSERV2 -f MQ_INSTALLATION_PATH/samp/amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.so \
       -r MQSeries_XA_RMI -s MPUT2:MPUT \
       -s MGET2:MGET \
       -v -bshm
$ buildclient -o doputs  -f MQ_INSTALLATION_PATH/samp/amqstxpx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.so
$ buildclient -o dogets  -f MQ_INSTALLATION_PATH/samp/amqstxgx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.so
```

5. Upravte soubor ubbstxcx.cfg a v případě potřeby přidejte podrobnosti o názvu počítače, pracovních adresářích a správci front:

```
$ tmloadcf    -y MQ_INSTALLATION_PATH/samp/ubbstxcx.cfg
```

6. Vytvořte TLOGDEVICE:

```
$tmadmin -c
```

Zobrazí se výzva k zadání. Na této obrazovce zadejte:

```
> crdl -z /<APPDIR>/TLOG1
```

7. Spusťte správce front:

```
$ strmqm
```

8. Spustit TUXEDO:

```
$ tmboot -y
```

Nyní můžete pomocí programů doputs a dogets vkládat zprávy do fronty a načítat je z fronty.

Sestavování prostředí serveru pro produkt WebSphere MQ for Windows (32bitový)

Poznámka: Změňte pole označená <> v následujících, na cesty k adresáři:

< MQMDIR>	cesta k adresáři uvedená při instalaci produktu WebSphere MQ , například `g:\Program Files\IBM\WebSphere MQ`
< TUXDIR>	cesta k adresáři uvedená při instalaci TUXEDO, například `f:\tuxedo`
< APPDIR>	cestu k adresáři, který má být použit pro ukázkovou aplikaci, například `f:\tuxedo\apps\mqapp`

Chcete-li sestavit prostředí serveru a ukázky, postupujte takto:

1. Vytvořte adresář aplikace, do kterého se má sestavit ukázková aplikace, například:

```
f:\tuxedo\apps\mqapp
```

2. Zkopírujte následující ukázkové soubory z ukázkového adresáře produktu WebSphere MQ do adresáře aplikace:

```
amqstxmn.mak
```

```
amqstxen.env
ubbstxcn.cfg
```

3. Upravte každý z těchto souborů a nastavte názvy adresářů a cesty k adresářům použité při instalaci.
4. Upravte soubor ubbstxcn.cfg (viz Obrázek 24 na stránce 158) a přidejte podrobnosti o názvu počítače a o správci front, ke kterému se chcete připojit.
5. Přidejte následující řádek do souboru TUXEDO < TUXDIR > udataobj\rm

```
MQSeries_XA_RMI;MQRMIXASwitchDynamic;
   <MQMDIR>\tools\lib\mqmxa.lib <MQMDIR>\tools\lib\mqm.lib
```

   kde < MQMDIR > je nahrazeno, jak je zobrazeno v předchozím příkladu. I když je zde zobrazen jako dva řádky, musí být nová položka v souboru jedna řádka.
6. Nastavte následující proměnné prostředí:

```
TUXDIR=<TUXDIR>
TUXCONFIG=<APPDIR>\tuxconfig
FIELDTBLS=<MQMDIR>\tools\c\samples\amqstxvx.fld
LANG=C
```

7. Vytvořte zařízení TLOG pro TUXEDO. Chcete-li to provést, vyvolejte příkaz `tmadmin -c` a zadejte příkaz:

```
crdl -z <APPDIR>\TLOG
```

   kde <*APPDIR*> je nahrazen.
8. Nastavte aktuální adresář na < APPDIR> a vyvolejte ukázkový soubor Makefile (amqstxmn.mak) jako soubor Makefile pro externí projekt. Například při použití jazyka Microsoft Visual C++ zadejte příkaz:

```
msvc amqstxmn.mak
```

   Vyberte volbu **sestavení** , chcete-li sestavit všechny ukázkové programy.

```
*RESOURCES
IPCKEY        99999
UID           0
GID           0
MAXACCESSERS  20
MAXSERVERS    20
MAXSERVICES   50
MASTER        SITE1
MODEL         SHM
LDBAL         N

*MACHINES
<MachineName> LMID=SITE1
              TUXDIR="f:\tuxedo"
              APPDIR="f:\tuxedo\apps\mqapp;g:\Program Files\IBM\WebSphere MQ\bin"
              ENVFILE="f:\tuxedo\apps\mqapp\amqstxen.env"
              TUXCONFIG="f:\tuxedo\apps\mqapp\tuxconfig"
              ULOGPFX="f:\tuxedo\apps\mqapp\ULOG"
              TLOGDEVICE="f:\tuxedo\apps\mqapp\TLOG"
              TLOGNAME=TLOG
              TYPE="i386NT"
              UID=0
              GID=0

*GROUPS
GROUP1
        LMID=SITE1  GRPNO=1
        TMSNAME=MQXA
        OPENINFO="MQSeries_XA_RMI:MYQUEUEMANAGER"

*SERVERS
DEFAULT: CLOPT="-A -- -m MYQUEUEMANAGER"

MQSERV1     SRVGRP=GROUP1 SRVID=1
MQSERV2     SRVGRP=GROUP1 SRVID=2

*SERVICES
MPUT1
MGET1
MPUT2
MGET2
```

Obrázek 24. Příklad souboru ubbstxcn.cfg pro produkt WebSphere MQ for Windows

Poznámka: Změňte názvy adresářů a cesty k adresářům tak, aby odpovídaly vaší instalaci. Změňte také název správce front MYQUEUEMANAGER na název správce front, ke kterému se chcete připojit. Další informace, které potřebujete přidat, jsou identifikovány pomocí znaků <> .

Ukázkový soubor ubbconfig pro produkt WebSphere MQ for Windows je uveden v tématu Obrázek 24 na stránce 158. Dodává se jako ubbstxcn.cfg v adresáři ukázek produktu WebSphere MQ .

Ukázkový soubor makefile (viz Obrázek 25 na stránce 159) dodaný pro produkt WebSphere MQ for Windows se nazývá ubbstxmn.maka je umístěn v adresáři ukázek produktu WebSphere MQ .

```
TUXDIR  = f:\tuxedo
MQMDIR  = g:\Program Files\IBM\WebSphere MQ
APPDIR  = f:\tuxedo\apps\mqapp
MQMLIB  = $(MQMDIR)\tools\lib
MQMINC  = $(MQMDIR)\tools\c\include
MQMSAMP = $(MQMDIR)\tools\c\samples
INC = -f "-I$(MQMINC) -I$(APPDIR)"
DBG = -f "/Zi"

amqstx.exe:
 $(TUXDIR)\bin\mkfldhdr    -d$(APPDIR) $(MQMSAMP)\amqstxvx.fld
 $(TUXDIR)\bin\viewc       -d$(APPDIR) $(MQMSAMP)\amqstxvx.v
 $(TUXDIR)\bin\buildtms    -o MQXA -r MQSeries_XA_RMI
 $(TUXDIR)\bin\buildserver -o MQSERV1 -f $(MQMSAMP)\amqstxsx.c \
                           -f $(MQMLIB)\mqm.lib -v $(INC) $(DBG) \
                           -r MQSeries_XA_RMI \
                           -s MPUT1:MPUT -s MGET1:MGET
 $(TUXDIR)\bin\buildserver -o MQSERV2 -f $(MQMSAMP)\amqstxsx.c \
                           -f $(MQMLIB)\mqm.lib -v $(INC) $(DBG) \
                           -r MQSeries_XA_RMI \
                           -s MPUT2:MPUT -s MGET2:MGET
 $(TUXDIR)\bin\buildclient -o doputs -f $(MQMSAMP)\amqstxpx.c \
                           -f $(MQMLIB)\mqm.lib -v $(INC) $(DBG)
 $(TUXDIR)\bin\buildclient -o dogets -f $(MQMSAMP)\amqstxgx.c \
                           -f $(MQMLIB)\mqm.lib $(INC) -v $(DBG)
 $(TUXDIR)\bin\tmloadcf    -y $(APPDIR)\ubbstxcn.cfg
```

Obrázek 25. Ukázkový soubor makefile TUXEDO pro produkt WebSphere MQ for Windows

Sestavování prostředí serveru pro produkt WebSphere MQ for Windows (64bitový)

Poznámka: Změňte pole označená <> v následujících, na cesty k adresáři:

< MQMDIR>	cesta k adresáři uvedená při instalaci produktu WebSphere MQ , například `g:\Program Files\IBM\WebSphere MQ`
< TUXDIR>	cesta k adresáři uvedená při instalaci TUXEDO, například `f:\tuxedo`
< APPDIR>	cestu k adresáři, který má být použit pro ukázkovou aplikaci, například `f:\tuxedo\apps\mqapp`

Chcete-li sestavit prostředí serveru a ukázky, postupujte takto:

1. Vytvořte adresář aplikace, do kterého se má sestavit ukázková aplikace, například:

   ```
   f:\tuxedo\apps\mqapp
   ```

2. Zkopírujte následující ukázkové soubory z ukázkového adresáře produktu WebSphere MQ do adresáře aplikace:

   ```
   amqstxmn.mak
   amqstxen.env
   ubbstxcn.cfg
   ```

3. Upravte každý z těchto souborů a nastavte názvy adresářů a cesty k adresářům použité při instalaci.
4. Upravte `ubbstxcn.cfg` (viz Obrázek 26 na stránce 160), abyste přidali podrobnosti o názvu počítače a správci front, ke kterému se chcete připojit.
5. Přidejte následující řádek do souboru TUXEDO <TUXDIR>`udataobj\rm`

   ```
   MQSeries_XA_RMI;MQRMIXASwitchDynamic;
      <MQMDIR>\tools\lib64\mqmxa64.lib <MQMDIR>\tools\lib64\mqm.lib
   ```

   kde < MQMDIR > je nahrazen. I když je zde zobrazen jako dva řádky, musí být nová položka v souboru jedna řádka.
6. Nastavte následující proměnné prostředí:

```
       TUXDIR=<TUXDIR>
       TUXCONFIG=<APPDIR>\tuxconfig
       FIELDTBLS=<MQMDIR>\tools\c\samples\amqstxvx.fld
       LANG=C
```

7. Vytvořte zařízení TLOG pro TUXEDO. Chcete-li to provést, vyvolejte příkaz `tmadmin -ca` zadejte příkaz:

   ```
   crdl -z <APPDIR>\TLOG
   ```

   kde <*APPDIR*> se nahradí, jak je uvedeno v předchozím příkladu.

8. Nastavte aktuální adresář na < APPDIR> a vyvolejte ukázkový soubor Makefile (amqstxmn.mak) jako soubor Makefile pro externí projekt. Například při použití jazyka Microsoft Visual C++ zadejte příkaz:

   ```
   msvc amqstxmn.mak
   ```

   Vyberte volbu **sestavení** , chcete-li sestavit všechny ukázkové programy.

```
*RESOURCES
IPCKEY        99999
UID           0
GID           0
MAXACCESSERS  20
MAXSERVERS    20
MAXSERVICES   50
MASTER        SITE1
MODEL         SHM
LDBAL         N

*MACHINES
<MachineName> LMID=SITE1
              TUXDIR="f:\tuxedo"
              APPDIR="f:\tuxedo\apps\mqapp;g:\Program Files\IBM\WebSphere MQ\bin"
              ENVFILE="f:\tuxedo\apps\mqapp\amqstxen.env"
              TUXCONFIG="f:\tuxedo\apps\mqapp\tuxconfig"
              ULOGPFX="f:\tuxedo\apps\mqapp\ULOG"
              TLOGDEVICE="f:\tuxedo\apps\mqapp\TLOG"
              TLOGNAME=TLOG
              TYPE="i386NT"
              UID=0
              GID=0

*GROUPS
GROUP1
        LMID=SITE1  GRPNO=1
        TMSNAME=MQXA
        OPENINFO="MQSeries_XA_RMI:MYQUEUEMANAGER"

*SERVERS
DEFAULT: CLOPT="-A -- -m MYQUEUEMANAGER"

MQSERV1     SRVGRP=GROUP1 SRVID=1
MQSERV2     SRVGRP=GROUP1 SRVID=2

*SERVICES
MPUT1
MGET1
MPUT2
MGET2
```

Obrázek 26. Příklad souboru ubbstxcn.cfg pro produkt WebSphere MQ for Windows

Poznámka: Změňte názvy adresářů a cesty k adresářům tak, aby odpovídaly vaší instalaci. Změňte také název správce front MYQUEUEMANAGER na název správce front, ke kterému se chcete připojit. Další informace, které potřebujete přidat, jsou identifikovány pomocí znaků <> .

Ukázkový soubor ubbconfig pro produkt WebSphere MQ for Windows je uveden v tématu Obrázek 26 na stránce 160. Dodává se jako ubbstxcn.cfg v adresáři ukázek produktu WebSphere MQ .

Ukázkový soubor makefile (viz Obrázek 27 na stránce 161) dodaný pro produkt WebSphere MQ for Windows se nazývá ubbstxmn.maka je umístěn v adresáři ukázek produktu WebSphere MQ .

```
TUXDIR  = f:\tuxedo
MQMDIR  = g:\Program Files\IBM\WebSphere MQ
APPDIR  = f:\tuxedo\apps\mqapp
MQMLIB  = $(MQMDIR)\tools\lib64
MQMINC  = $(MQMDIR)\tools\c\include
MQMSAMP = $(MQMDIR)\tools\c\samples
INC = -f "-I$(MQMINC) -I$(APPDIR)"
DBG = -f "/Zi"

amqstx.exe:
 $(TUXDIR)\bin\mkfldhdr    -d$(APPDIR) $(MQMSAMP)\amqstxvx.fld
 $(TUXDIR)\bin\viewc       -d$(APPDIR) $(MQMSAMP)\amqstxvx.v
 $(TUXDIR)\bin\buildtms    -o MQXA -r MQSeries_XA_RMI
 $(TUXDIR)\bin\buildserver -o MQSERV1 -f $(MQMSAMP)\amqstxsx.c \
                           -f $(MQMLIB)\mqm.lib -v $(INC) $(DBG) \
                           -r MQSeries_XA_RMI \
                           -s MPUT1:MPUT -s MGET1:MGET
 $(TUXDIR)\bin\buildserver -o MQSERV2 -f $(MQMSAMP)\amqstxsx.c \
                           -f $(MQMLIB)\mqm.lib -v $(INC) $(DBG) \
                           -r MQSeries_XA_RMI \
                           -s MPUT2:MPUT -s MGET2:MGET
 $(TUXDIR)\bin\buildclient -o doputs -f $(MQMSAMP)\amqstxpx.c \
                           -f $(MQMLIB)\mqm.lib -v $(INC) $(DBG)
 $(TUXDIR)\bin\buildclient -o dogets -f $(MQMSAMP)\amqstxgx.c \
                           -f $(MQMLIB)\mqm.lib $(INC) -v $(DBG)
 $(TUXDIR)\bin\tmloadcf    -y $(APPDIR)\ubbstxcn.cfg
```

Obrázek 27. Ukázkový soubor makefile TUXEDO pro produkt WebSphere MQ for Windows

Ukázkový serverový program pro TUXEDO

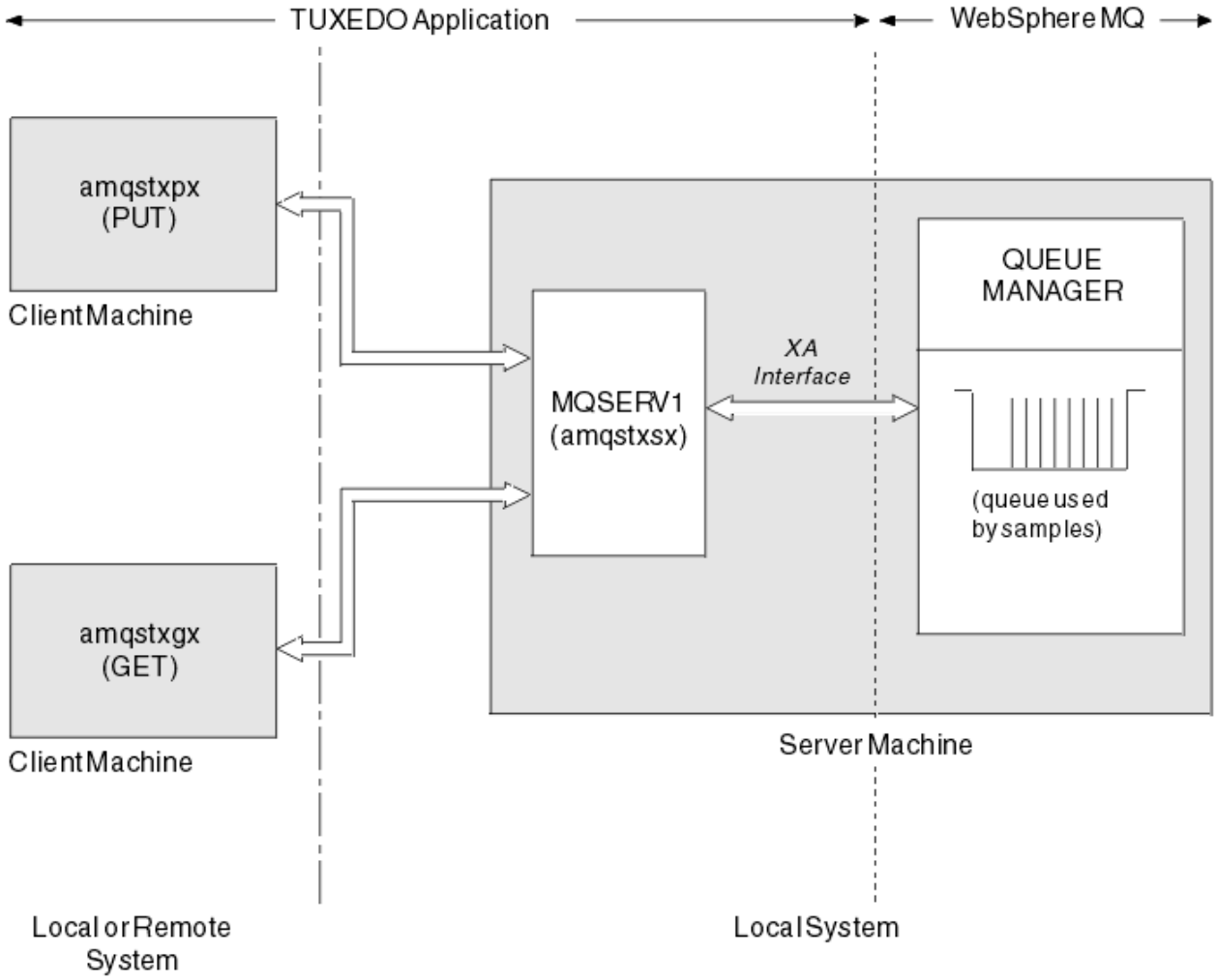
Ukázkový program serveru (amqstxsx) je navržen pro práci s ukázkovými programy Put (amqstxpx.c) a Get (amqstxgx.c). Ukázkový program serveru se spustí automaticky, když se spustí TUXEDO.

Poznámka: Musíte spustit správce front ***před*** spuštěním TUXEDO.

Ukázkový server poskytuje dvě služby TUXEDO, MPUT1 a MGET1:

- Služba MPUT1 je řízena ukázkou PUT a používá příkaz MQPUT1 v synchronizačním bodu k vložení zprávy do jednotky práce, kterou řídí TUXEDO. Vezme parametry QName a Text zprávy, které jsou dodány ukázkou PUT.
- Služba MGET1 se otevře a zavře frontu pokaždé, když získá zprávu. Vezme parametry QName a Text zprávy, které jsou dodány ukázkou GET.

Všechny chybové zprávy, kódy příčiny a stavové zprávy jsou zapsány do souboru protokolu TUXEDO.



Obrázek 28. Jak se vzorky TUXEDO vzájemně spolupracují

Vložit vzorový program pro TUXEDO

Tato ukázka vám umožňuje vložit zprávu do fronty vícekrát, v dávkách, demonstrovat pomocí protokolu TUXEDO jako správce prostředků syncpointing.

Ukázkový program serveru `amqstxsx` musí být spuštěn, aby byl ukázkový ukázkový soubor úspěšný; ukázkový program serveru se připojí ke správci front a použije rozhraní XA. Chcete-li spustit ukázku, zadejte:

- `doputs -n queuename -b batchsize -c trancount -t message`

Příklad:

- `doputs -n myqueue -b 5 -c 6 -t "Hello World"`

To vloží 30 zpráv do fronty s názvem `myqueue`, v šesti dávkách, každý s pěti zprávami v něm. Pokud dojde k problémům, zazálohuje dávku zpráv, jinak je potvrdí.

Jakékoli chybové zprávy jsou zapsány do souboru protokolu TUXEDO a na stderr. Jakékoli kódy příčiny jsou zapisovány do stderr.

Získat ukázku pro TUXEDO

Tato ukázka vám umožňuje získat zprávy z fronty v dávkách.

Ukázkový program serveru `amqstxsx` musí být spuštěn, aby byl ukázkový ukázkový soubor úspěšný; ukázkový program serveru se připojí ke správci front a použije rozhraní XA. Chcete-li spustit ukázku, zadejte:

- `dogets -n queuename -b batchsize -c trancount`

Příklad:

- `dogets -n myqueue -b 6 -c 4`

To zabere 24 zpráv z fronty s názvem `myqueue`, v šesti dávkách, každá se čtyřmi zprávami v ní. Spustíte-li tento příkaz po vložení příkladu, který vloží 30 zpráv v systému `myqueue`, máte na serveru `myqueuepouze` šest zpráv. Počet dávek a velikost dávky se mohou lišit mezi vložením zpráv a získáním těchto dávek.

Jakékoli chybové zprávy jsou zapsány do souboru protokolu TUXEDO a na stderr. Jakékoli kódy příčiny jsou zapisovány do stderr.

Použití uživatelské procedury zabezpečení SSPI v systémech Windows

Toto téma popisuje, jak používat programy pro ukončení kanálů SSPI v systémech Windows . Dodaný kód ukončení je ve dvou formátech: objekt a zdroj.

Kód objektu

Soubor s kódem objektu se nazývá amqrspin.dll. Pro klienta i server je instalován jako standardní součást produktu WebSphere MQ for Windows ve složce *MQ_INSTALLATION_PATH*`/exits/`*INSTALLATION_NAME* . Například `C:\Program Files\IBM\WebSphere MQ\exits\installation2`. Je načten jako standardní uživatelský vstup. Předaný konec kanálu zabezpečení můžete spustit a použít ověřovací služby ve své definici kanálu.

Chcete-li to provést, zadejte jednu z následujících možností:

```
SCYEXIT('amqrspin(SCY_KERBEROS)')

SCYEXIT('amqrspin(SCY_NTLM)')
```

Chcete-li poskytnout podporu pro omezený kanál, zadejte na kanálu SVRCONN následující:

```
SCYDATA('remote_principal_name')
```

kde *název_vzdáleného_činitele* je ve tvaru DOMAIN\uživatel. Zabezpečený kanál je vytvořen pouze v případě, že se název vzdáleného činitele shoduje s názvem *název_vzdáleného_činitele*.

Chcete-li mezi systémy, které jsou provozovány v doméně zabezpečení Kerberos , používat dodávaný kanál s ukončovacími programy, vytvořte pro správce front název servicePrincipal.

Zdrojový kód

Soubor zdrojového kódu ukončení se nazývá amqsspin.c. Nachází se v `C:\Program Files\IBM\WebSphere MQ\Tools\c\Samples`.

Změníte-li zdrojový kód, musíte upravený zdroj znovu zkompilovat.

Kompilujete a propojíte jej stejným způsobem jako ostatní uživatelské procedury kanálu pro příslušnou platformu, kromě toho, že záhlaví SSPI musí být přístupná v době kompilace a knihovny zabezpečení SSPI spolu s doporučenými přidruženými knihovnami je třeba k nim přistupovat v době propojení.

Než spustíte následující příkaz, ujistěte se, že cl.exea knihovna Visual C++ a složka `include` jsou dostupné ve vaší cestě. Příklad:

```
cl /VERBOSE /LD /MT /I<path_to_Microsoft_platform_SDK\include>
/I<path_to_WebSphere MQ\tools\c\include> amqsspin.c /DSECURITY_WIN32
-link  /DLL /EXPORT:SCY_KERBEROS /EXPORT:SCY_NTLM STACK:8192
```

Poznámka: Zdrojový kód neobsahuje žádné ustanovení pro trasování nebo ošetření chyb. Pokud upravíte a použijete zdrojový kód, přidejte své vlastní trasovací a manipulační rutiny.

Spuštění ukázek pomocí vzdálených front

Vzdálené řazení do front lze demonstrovat spuštěním ukázek v připojených správcích front.

Program amqscos0.tst poskytuje lokální definici vzdálené fronty (SYSTEM.SAMPLE.REMOTE), který používá vzdáleného správce front s názvem OTHER. Chcete-li použít tuto definici ukázky, změňte hodnotu OTHER na název druhého správce front, který chcete použít. Musíte také nastavit kanál zpráv mezi dvěma správci front; chcete-li získat informace o tom, jak to provést, prostudujte si téma Definování kanálů.

Vzorové programy požadavku umístí své vlastní lokální názvy správce front do pole *ReplyToQMgr* zpráv, které odesílají. Funkce Inquire and Set slouží k odeslání zpráv odpovědi do fronty a správce front zpráv pojmenované v polích *ReplyToQ* a *ReplyToQMgr* v rámci zpráv požadavků, které zpracovávají.

Ukázkový program pro monitorování front klastru (AMQSCLM)

Tato ukázka používá vestavěné funkce vyrovnávání pracovní zátěže klastru IBM WebSphere MQ pro přímé zprávy do instancí front, které mají připojené aplikace. Tento automatický směr zabraňuje sestavení zpráv na instanci fronty klastru, ke které není připojena žádná spotřeba aplikace.

Přehled

Pro stejnou frontu v různých správcích front můžete nastavit klastr, který má více než jednu definici téže fronty. Tato konfigurace poskytuje výhodu zvýšené dostupnosti a vyrovnávání pracovní zátěže. Nicméně v produktu IBM WebSphere MQ není vestavěna žádná schopnost dynamicky upravovat distribuci zpráv v klastru, a to na základě stavu připojených aplikací. Z tohoto důvodu musí být aplikace spotřebitele vždy připojena ke každé instanci fronty, aby bylo zajištěno, že zprávy budou zpracovány.

Ukázkový program monitorování fronty klastru monitoruje stav připojených aplikací. Program dynamicky upravuje konfiguraci vyrovnávání pracovní zátěže pro přímé zprávy na instance klastrované fronty s připojenými aplikacemi spotřebovávající spotřebu. V určitých situacích lze tento program použít k uvolnění nutnosti připojení aplikace tak, aby vždy byla připojena ke každé instanci fronty. Také znovu odešle zprávy, které budou zařazeny do fronty na instanci fronty bez připojených přijímajících aplikací. Opětovné odeslání zpráv umožňuje směrovat zprávy do oblasti spotřebovávající aplikaci, která je dočasně ukončena.

Program je navržen tak, aby jej bylo možné použít tam, kde náročné aplikace jsou dlouhodobě spuštěné aplikace, spíše než často připojovaná a odpojující aplikace.

Ukázkový program pro monitorování fronty klastru je kompilovaným spustitelným programem v ukázkovém souboru C `amqsclma.c`.

Další informace o klastrech a pracovní zátěži naleznete v tématu Použití klastrů pro správu pracovní zátěže .

AMQSCLM: Návrh a plánování pro použití ukázky

Informace o tom, jak pracuje ukázkový program monitorování fronty klastru, je třeba zvážit při nastavení systému pro spuštění ukázkového programu a o úpravách, které lze provést v ukázkovém zdrojovém kódu.

Návrh

Ukázkový program monitorování fronty klastru monitoruje lokální klastrované fronty, které mají připojené aplikace. Program monitoruje fronty určené uživatelem. Název fronty může být specifický, například `APP.TEST01`, nebo generický. Generické názvy musí být ve formátu, který je v souladu s PCF (Programmable Command Format). Příklady generických názvů jsou `APP.TEST*`, nebo `APP*`.

Každý správce front v klastru, který vlastní instanci lokální fronty, která má být monitorována, vyžaduje k připojení instanci ukázkového programu monitorování fronty klastru.

Dynamické směrování zpráv

Ukázkový program monitorování front klastru používá hodnotu fronty **IPPROCS** (otevřeno pro výpočet vstupních procesů) fronty k určení, zda má tato fronta nějaké odběratele. Hodnota větší než 0 označuje, že

fronta má připojenu alespoň jednu odebírající aplikaci. Takové fronty jsou aktivní. Hodnota 0 znamená, že fronta nemá žádné připojené náročné programy. Takové fronty jsou neaktivní.

U klastrované fronty s více instancemi v klastru produkt WebSphere MQ používá vlastnost priority zátěže klastru **CLWLPRTY** každé instance fronty k určení instancí, do kterých mají být odesílány zprávy. Produkt WebSphere MQ odesílá zprávy do dostupných instancí fronty s nejvyšší hodnotou **CLWLPRTY** .

Ukázkový program pro monitorování fronty klastru aktivuje frontu klastru nastavením lokální hodnoty **CLWLPRTY** na hodnotu 1. Program deaktivuje frontu klastru nastavením jeho hodnoty **CLWLPRTY** na hodnotu 0.

Technologie klastrování produktu WebSphere MQ šíří aktualizovanou vlastnost produktu **CLWLPRTY** klastrované fronty ke všem relevantním správcům front v klastru. Například

- Správce front s připojenou aplikací, která vkládá zprávy do fronty.
- Správce front, který vlastní lokální frontu se stejným názvem ve stejném klastru.

Šíření se provádí pomocí správců front úplného úložiště klastru. Nové zprávy pro frontu klastru jsou směrovány na instance s nejvyšší hodnotou **CLWLPRTY** v rámci klastru.

Přenos zprávy ve frontě

Dynamická úprava hodnoty **CLWLPRTY** ovlivňuje směrování nových zpráv. Tato dynamická úprava neovlivňuje zprávy, které jsou již ve frontě na instanci fronty bez připojených spotřebitelů, nebo zprávy, které byly přes mechanizmus vyrovnávání pracovní zátěže před šířením upravené hodnoty **CLWLPRTY** , byly propagovány v rámci klastru. V důsledku toho zůstanou zprávy v žádné neaktivní frontě a nebudou zpracovány aplikací spotřebovávající spotřebu. K vyřešení tohoto problému je ukázkový program pro monitorování fronty klastru schopen získat zprávy z lokální fronty bez spotřebitelů a odeslat tyto zprávy vzdáleným instancím stejné fronty, kde jsou spotřebitelé připojeni.

Ukázkový program monitorování fronty klastru přenáší zprávy z neaktivní lokální fronty do jedné nebo více aktivních vzdálených front tím, že získává zprávy (pomocí příkazu **MQGET**) a vkládá zprávy (pomocí **MQPUT**) do stejné klastrované fronty. Tento přenos způsobí, že správa pracovní zátěže klastru produktu WebSphere MQ vybere jinou cílovou instanci založenou na vyšší hodnotě CLWLPRTY , než je hodnota instance lokální fronty. Perzistence a kontext zprávy jsou během přenosu zprávy zachovány. Pořadí zpráv a všechny volby vazby nejsou zachovány.

Naplánování

Ukázkový program monitorování fronty klastru upraví konfiguraci klastru, když dojde ke změně v konektivitě aplikací odběratele. Změny jsou přeneseny ze správců front, ve kterých je ukázkový program pro monitorování fronty klastru, do úplných správců front úložiště v klastru. Správci front úplného úložiště zpracují aktualizace konfigurace a znovu je odešlou do všech příslušných správců front v klastru. Relevantní správci front zahrnují tyto správce front, kteří vlastní klastrované fronty se stejným názvem (kde je spuštěna instance ukázkového programu pro monitorování fronty klastru) a správce front, v němž aplikace otevřela frontu klastru k vložení zpráv do fronty za posledních 30 dní.

Změny jsou asynchronně zpracovány v celém klastru. Proto mohou různí správci front v klastru po každé změně mít různé pohledy na konfiguraci po určitou dobu.

Ukázkový program pro monitorování fronty klastru je vhodný pouze pro systémy, kde se spotřebovávají aplikace zřídka připojují nebo odpojují; například dlouho běžící náročné aplikace. Když se používá k monitorování systémů, kde jsou náročné aplikace připojeny pouze pro krátká období, může latence při distribuci aktualizací konfigurace vést k tomu, že správci front v klastru mají nesprávný pohled na fronty, kde jsou spotřebitelé připojeni. Tato latence může vést k nesprávně přesměrovaným zprávám.

Při monitorování mnoha front může relativně nízká míra změn u připojených spotřebitelů ve všech frontách zvýšit provoz konfigurace klastru v rámci klastru. Zvýšený provoz konfigurace klastru může mít za následek nadměrné zatížení na jednom nebo více následujících správcích front.

- Správci front, ve kterém je spuštěn ukázkový program pro monitorování fronty klastru
- Správci front úplného úložiště

- Správce front s připojenou aplikací, která vkládá zprávy do fronty.
- Správce front, který vlastní lokální frontu se stejným názvem ve stejném klastru.

Musí být posouzeno použití procesoru na správci front úplného úložiště. Další využití procesoru je viditelné jako provoz zpráv v úplné frontě úložišť SYSTEM.CLUSTER.COMMAND.QUEUE. Pokud se zprávy v této frontě sestavují, znamená to, že správci front úplného úložiště nejsou schopni udržet krok s rychlostí změny konfigurace klastru v systému.

Při monitorování mnoha front pomocí ukázkového programu pro monitorování fronty klastru je k dispozici množství práce provedené ukázkovým programem a správcem front. Tato práce se provádí i v případě, kdy nejsou k dispozici žádné změny u připojených spotřebitelů. Argument **-i** lze upravit tak, aby se snížilo využití procesoru vzorového programu na lokálním systému, a to snížením frekvence monitorovacího cyklu.

Ukázkový program monitorování fronty klastru pomáhá detekovat nadměrné aktivity a uvádí průměrnou dobu zpracování na interval zjišťování, uplynulý čas zpracování a počet změn konfigurace. Sestavy se doručí v informační zprávě, **CLM0045I**, každých 30 minut nebo každých 600 dotazových intervalů, podle toho, co nastane dříve.

Požadavky na použití monitorování fronty klastru

Ukázkový program monitorování fronty klastru má požadavky a omezení. Vzorový zdrojový kód poskytnutý pro změnu některých těchto omezení můžete upravit tak, jak jej lze použít. Příklady uvedené v podrobných úpravách této sekce, které lze provést.

- Ukázkový program pro monitorování fronty klastru je navržen tak, aby jej bylo možné použít k monitorování front, ve kterých jsou buď připojeny aplikace, které jsou připojeny, nebo nejsou připojeny. Pokud systém spotřebovává aplikace, které jsou často připojovaných a odpojování, může ukázkový program generovat nadměrnou aktivitu konfigurace klastru v celém klastru. To může mít vliv na výkon správců front v klastru.
- Ukázkový program monitorování fronty klastru závisí na použité systémové a klastrované technologii produktu WebSphere MQ . Počet monitorovaných front, četnost monitorování a četnost změn stavu každé fronty ovlivní zatížení na celkovém systému. Tyto faktory je třeba vzít v úvahu při výběru front, které mají být monitorovány, a intervalu výzev k monitorování.
- Instance ukázkového programu pro monitorování fronty klastru musí být připojena ke každému správci front v klastru, který vlastní instanci fronty, která má být monitorována. Není nutné připojit ukázkový program ke správcům front v klastru, který nevlastní fronty.
- Ukázkový program monitorování fronty klastru musí být spuštěn s vhodnou autorizací pro přístup ke všem vyžadovaným prostředkům produktu WebSphere MQ . Například
  - Správce front, k němuž má být připojen
  - SYSTEM.ADMIN.COMMAND.QUEUE
  - Všechny fronty, které mají být monitorovány, když se provádí přenos zpráv
- Příkazový server musí být spuštěn pro každého správce front se připojeným ukázkovým programem pro monitorování fronty klastru.
- Každá instance ukázkového programu pro monitorování front klastru vyžaduje výlučné použití lokální fronty (neklastrované) ve správci front, k němuž je připojen. Tato lokální fronta se používá k řízení ukázkového programu a přijímá zprávy odpovědí z inquires vytvořených na příkazový server správce front.
- Všechny fronty, které mají být monitorovány jedinou instancí ukázkového programu monitorování fronty klastru, musí být ve stejném klastru. Má-li správce front fronty ve více klastrech, které vyžadují monitorování, je zapotřebí více instancí ukázkového programu. Každá instance potřebuje lokální frontu pro řídicí a odpovědní zprávy.
- Všechny fronty, které mají být monitorovány, musí být v jednom klastru. Fronty konfigurované pro použití seznamu názvů klastru nejsou monitorovány.

- Povolení přenosu zpráv z neaktivních front je volitelné. Vztahuje se na všechny fronty monitorované instancí ukázkového programu monitorování fronty klastru. Pokud je povolena pouze podmnožina monitorovaných front, je třeba přenos dvou instancí programu pro monitorování fronty klastru, který je povolen. Jeden ukázkový program má povolen přenos zpráv a druhý přenos zprávy je zakázán. Každá instance ukázkového programu potřebuje lokální frontu pro řídicí a odpovědní zprávy.
- WebSphere MQ -vyrovnávání pracovní zátěže klastru bude standardně odesílat zprávy instancím klastrovaných front, které jsou umístěny ve stejném správci front, ke kterému je aplikace připojena. Tato volba musí být zakázána, pokud je lokální fronta neaktivní za následujících okolností:
  - Aplikace aplikací se připojí ke správcům front, kteří vlastní instance neaktivní fronty, která je monitorována.
  - Zprávy ve frontě jsou převáděny z neaktivních front do aktivních front.

  Lokální preference vyrovnávání pracovní zátěže ve frontě lze zakázat staticky, a to nastavením hodnoty CLWLUSEQ na hodnotu ANY. V těchto konfiguračních zprávách jsou lokální fronty distribuovány do lokálních a vzdálených instancí front pro vyvážení pracovní zátěže, a to i v případě, kdy existují lokální náročné aplikace. Ukázkový program monitorování fronty klastru může být také konfigurován tak, aby dočasně nastavil hodnotu **CLWLUSEQ** na ANY , zatímco fronta nemá žádné připojené spotřebitele, což vede k tomu, že lokální zprávy budou v době, kdy jsou tyto fronty aktivní, odesílány pouze lokální zprávy.
- Systém WebSphere MQ a aplikace nesmějí používat produkt **CLWLPRTY** pro monitorované fronty nebo kanály, které mají být používány. Jinak by akce ukázkového programu monitorování fronty klastru na atributech fronty produktu **CLWLPRTY** mohly mít nepožadované účinky.
- Ukázkový program pro monitorování fronty klastru protokoluje informace o běhovém prostředí do sady souborů sestav. Požaduje se adresář pro uložení těchto sestav a ukázkový program pro monitorování fronty klastru musí mít oprávnění k zápisu do tohoto adresáře.

AMQSCLM: Příprava a spuštění ukázky

Chcete-li spustit ukázku monitorování fronty klastru, musíte nakonfigurovat správce front tak, aby bezpečně přijímal příchozí požadavky na připojení z aplikací spuštěných v režimu klienta.

Než začnete

Před spuštěním ukázky monitorování fronty klastru je třeba provést následující kroky.

1. Vytvořte pracovní frontu v každém správci front za účelem interního použití ukázky.

   Každá instance ukázky potřebuje lokální neklastrovou frontu pro výlučné interní použití. Můžete zvolit název fronty. Příklad používá název AMQSCLM.CONTROL.QUEUE. Například v systému Windowsmůžete tuto frontu vytvořit pomocí příkazu **MQSC** .

   ```
   DEFINE QLOCAL(AMQSCLM.CONTROL.QUEUE)
   ```

   Hodnoty **MAXDEPTH** a **MAXMSGL** můžete ponechat jako výchozí.

2. Vytvořte adresář pro protokoly chyb a informací o chybách.

   Ukázka vypíše diagnostické zprávy do souborů sestav. Je třeba vybrat adresář, do kterého chcete soubory uložit. V systému Windowsmůžete například vytvořit adresář pomocí následujícího příkazu:

   ```
   mkdir C:\AMQSCLM\rpts
   ```

   Soubory sestavy vytvořené v ukázce mají následující konvenci pojmenování:

   ```
   QmgrName.ClusterName.RPT0n.LOG
   ```

3. (Volitelné) Definujte ukázku monitorování fronty klastru jako službu IBM WebSphere MQ .

   Chcete-li monitorovat fronty, musí být vzorek vždy spuštěn. Chcete-li se ujistit, že je ukázka monitorování fronty klastru vždy spuštěna, můžete definovat ukázku jako službu správce front. Definování ukázky jako služby znamená, že je spuštěn příkaz AMQSCLM při spuštění správce front.

K definování ukázky monitorování fronty klastru jako služby IBM WebSphere MQ můžete použít následující příklad příkazu **RUNMQSC** .

```
define service(AMQSCLM) +
    descr('Active Cluster Queue Message Distribution Monitor - AMQSCLM') +
    control(qmgr) +
    servtype(server) +
    startcmd('<Install Root>\tools\c\samples\Bin\AMQSCLM.exe') +
 startarg('-m +QMNAME+ -c CLUSTER1 -q ABC* -r AMQSCLM.CONTROL.QUEUE -l c:\AMQSCLM\rpts') +
 stdout('C:\AMQSCLM\rpts\+QMNAME+.TSTCLUS.stdout.log') +
 stderr('C:\AMQSCLM\rpts\+QMNAME+.TSTCLUS.stderr.log')
```

kde < Install Root > je umístění vaší instalace.

Definice	Popis
service	Uvádí název služby. Můžete zvolit název služby.
descr	Uvádí textový popis služby.
control	Označuje, že služba se spustí a zastaví ve stejnou dobu jako správce front.
servtype	Označuje, že objekt služby serveru, což znamená pouze jednu instanci, může být proveden v daném okamžiku pro tohoto správce front.
startcmd	Uvádí umístění a jméno programu.
startarg	Určuje argumenty ukázky. Všimněte si použití volby *+ QMNAME +*. Název správce front se automaticky nahrazuje názvem správce front.
stdout	Plně kvalifikovaný název souboru, do kterého je přesměrován standardní výstup. Ukázka zapisuje do tohoto souboru pouze zprávy potvrzující, že vzorek byl ukončen. Ukázka to provede, protože soubor se standardním chybovým souborem již byl uzavřen v předchozí fázi ukázkového procesu ukončení.
stderr	Úplný název souboru, na který je přesměrován standardní výstup chybových hlášení. Ukázka zapisuje na standardní chybový soubor všechny chybové zprávy před ukončením ukázky.

Informace o této úloze

Tato úloha vám umožňuje spustit a zastavit ukázku monitorování fronty klastru různými způsoby. Umožňuje vám také spustit ukázku v režimu, který generuje soubory sestav obsahující statistické informace o monitorovaných frontách.

Ukázkový program lze spustit pomocí následujícího příkazu.

```
AMQSCLM -m QMgrName -c ClusterName (-q QNameMask | -f QListFile) -r MonitorQName
[-l ReportDir] [-t] [-u ActiveVal] [-i Interval] [-d] [-s] [-v]
```

V tabulce jsou uvedeny argumenty, které lze použít spolu s ukázkou monitorování fronty klastru, spolu s dalšími informacemi o každém z nich.

Argument	Proměnná	Další informace
-m	**QMgrName**	Správce front, který má být sledován.
-c	**ClusterName**	Klastr obsahující fronty určené k monitorování.
-q	**QNameMask**	Fronta, nebo fronty k monitorování. Koncový * monitoruje všechny fronty s názvy, které odpovídají nula nebo více koncových znacích.
-f	**QListFile**	Úplná cesta a název souboru obsahujícího seznam názvů front pro masky názvu fronty názvů, které mají být monitorovány. Soubor musí obsahovat jeden název fronty/masku na řádek. Můžete uvést **-q** nebo **-f**, ale ne obojí.
-r	**MonitorQName**	Lokální fronta je používána exkluzivně pro vzorek.
-l	**ReportDir**	Cesta k adresáři, do kterého mají být ukládány protokolované informační zprávy v sadě zalamování < fn> Pro každou kombinaci správce front a fronty je generován soubor sestav, který je s omezením na určitou velikost. Zapisovač protokolu vždy zapisuje do stejného souboru, ale uchovává také dvě předchozí verze souboru. < /fn> soubory sestav.
-t		(Volitelné) Umožňuje přenos zpráv ve frontě z neaktivních lokálních front do aktivních front. Není-li tato možnost povolena, jsou do aktivních instancí fronty dynamicky směrovány pouze nové zprávy, které vstupují do klastru.
-u	**ActiveVal**	(Volitelné) Automaticky přepíná vlastnost **CLWLUSEQ** monitorované instance fronty na ANY , pokud je neaktivní, a na vlastnost **ActiveVal** , pokud je aktivní. **ActiveVal** může být LOCAL nebo QMGR. Není-li tento argument nastaven v systému, kde jsou aplikace připojeny ke stejnému správci front nebo je-li přenos zpráv povolen, pak monitorované fronty musí mít hodnotu **CLWLUSEQ** ANYnebo QMGR s hodnotou ANYsprávce front.
-i	**Interval**	(Volitelné) Časový interval v sekundách, ve kterém monitor kontroluje fronty. Výchozí hodnota je 300 sekund (5 minut).
-d		(Volitelné) Umožňuje další diagnostický výstup. Ladicí výstup může být užitečný při počáteční konfiguraci systému nebo při práci s ukázkovým kódem.
-s		(Volitelné) Povolí minimální statistický výstup na každý interval.
-v		(Volitelné) Chcete-li kromě souborů sestav také protokolovat informace o sestavě do produktu `standard out`.

Příklady seznamu argumentů:

```
-m QMGR1 -c CLUS1 -f c:\QList.txt -r CLMQ -l c:\amqsclm\rpts -s
-m QMGR2 -c CLUS1 -q ABC* -r CLMQ -l c:\amqsclm\rpts -i 600
-m QMGR1 -c CLUSDEV -q QUEUE.* -r CLMQ -l c:\amqsclm\rpts -t -u QMGR -d
```

Ukázkový soubor se seznamem front:

```
Q1
QUEUE.*
ABC
ABD
```

Postup

1. Spusťte ukázku monitorování fronty klastru. Ukázku můžete spustit jedním z následujících způsobů:

- Použijte příkazový řádek s odpovídajícími oprávněními uživatele.
- Pokud je ukázka konfigurována jako služba IBM WebSphere MQ , použijte příkaz MQSC **START SERVICE** .

Seznam argumentů je v obou případech stejný.

Ukázka nespouští monitorování front po dobu 10 sekund od inicializace programu. Toto zpoždění umožňuje náročným aplikacím nejprve se připojit k monitorovaným frontám, čímž se zabrání zbytečným změnám aktivního stavu fronty.

2. Zastavte ukázku monitorování fronty klastru. Ukázka se automaticky zastaví při zastavení činnosti správce front, zastavení, uvádění do klidového stavu nebo v případě, že je připojení ke správci front přerušeno. Existují způsoby, jak zastavit ukázku bez ukončení správce front:
   - Nakonfigurujte lokální frontu použitou výhradně ukázkou k zakázání funkce Get.
   - Odešlete zprávu s **CorrelId** z `"STOP CLUSTER MONITOR\0\0\0\0"`do lokální fronty použité výlučně pro ukázku.
   - Ukončete ukázkový proces. To může vést ke ztrátě netrvalých zpráv přenášených do aktivních front. Může to také vést k tomu, že lokální fronta použitá vzorkovým vzorkem je otevřená po dobu několika sekund po ukončení. Tato situace zabrání okamžitému spuštění nové instance ukázky monitorování fronty klastru.

   Pokud byl vzorek spuštěn jako služba IBM WebSphere MQ , produkt **STOP SERVICE** nemá žádný efekt. Je možné použít jednu z metod ukončení popsaných jako konfigurovaný mechanizmus **STOP SERVICE** ve správci front.

Jak pokračovat dále

Zkontrolujte stav ukázky.

Je-li vykazování povoleno, můžete přezkoumat soubory sestavy pro stav. Použijte následující příkaz k přezkoumání nejnovějšího souboru sestavy.

```
QMgrName.ClusterName.RPT01.LOG
```

Chcete-li zkontrolovat starší soubory sestav, použijte následující příkazy.

```
QMgrName.ClusterName.RPT02.LOG
QMgrName.ClusterName.RPT03.LOG
```

Velikost souborů sestavy vzroste na maximální velikost přibližně 1 MB. Když se soubor RPT01 zaplní, vytvoří se nový soubor RPT01 . Starý soubor RPT01 je přejmenován na RPT02. RPT02 je přejmenován na RPT03. Původní RPT03 je vyřazeno.

Ukázka vytváří informační zprávy v následujících situacích:

- Při spuštění
- Při ukončení
- Když je zaznačit frontu **ACTIVE** nebo **INACTIVE**
- když znovu požaduje zprávy z neaktivní fronty k aktivní instanci nebo instancím

Ukázka vytvoří chybovou zprávu *CLMnnnnE* k ohlášení problému, který vyžaduje pozornost.

Každých 30 minut vzorková zpráva uvádí průměrnou dobu zpracování na interval zjišťování a uplynulý čas zpracování. Tyto informace jsou uchovávány ve zprávě CLM0045I.

Jsou-li povoleny statistické zprávy **-s**, sestava ukázek vykazuje následující statistické informace o každé kontrole fronty:

- Čas potřebný ke zpracování front (v milisekundách)
- Počet zkontrolovaných front
- Počet aktivních/neaktivních změn provedených

- Počet přenesených zpráv

Tyto informace jsou uvedeny ve zprávě CLM0048I.

Soubory sestavy mohou rychle růst v režimu ladění a rychle se zalomit. V této situaci může být překročen limit velikosti 1-MB pro jednotlivé soubory.

AMQSCLM: Odstraňování problémů

Následující sekce obsahují informace o scénářích, které mohou být zjištěny při použití ukázky. Poskytují se informace o možných vysvětleních scénáře a o možnostech řešení tohoto scénáře.

Scénář: Produkt AMQSCLM se nespustí

Potenciální vysvětlení: Chybná syntaxe.

Akce: Zkontrolujte standardní chybový výstup pro správnou syntaxi

Potenciální vysvětlení: Správce front není k dispozici.

Akce: Zkontrolujte soubor sestavy pro ID zprávy CLM0010E.

Potenciální vysvětlení: Nelze otevřít nebo vytvořit soubor sestavy nebo soubory.

Akce: Během inicializace zkontrolujte chybové zprávy standardního výstupu chyb.

Scénář: AMQSCLM nemění frontu na AKTIVNÍ nebo NEAKTIVNÍ

Potenciální vysvětlení: Fronta se nenachází v seznamu front, které mají být monitorovány.

Akce: Zkontrolujte hodnoty parametrů **-q** a **-f** .

Potenciální vysvětlení: Fronta není lokální frontou ve správném klastru.

Akce: Zkontrolujte, zda je fronta lokální a ve správném klastru.

Potenciální vysvětlení: AMQSCLM není spuštěn pro tohoto správce front a klastru.

Akce: Spusťte AMQSCLM pro příslušného správce front a klastru.

Potenciální vysvětlení: Fronta je ponechána NEAKTIVNÍ, **CLWLPRTY**= 0, protože nemá žádné spotřebitele. Případně je ponechán AKTIVNÍ **CLWLPRTY**> =1, protože má alespoň 1 odběratele.

Akce: Zkontrolujte, zda jsou k frontě připojeny náročné aplikace.

Potenciální vysvětlení: Příkazový server správce front není spuštěn.

Akce: Zkontrolujte, zda soubory sestav nejsou chyby.

Scénář: Zprávy nejsou směrovány kolem NEAKTIVNÍ fronty.

Potenciální vysvětlení: Zprávy jsou vloženy přímo do správce front, který vlastní neaktivní frontu, a hodnota **CLWLUSEQ** fronty není ANYa argument **-u** se pro AMQSCLM nepoužívá.

Akce: Zkontrolujte hodnotu proměnné **CLWLUSEQ** příslušného správce front nebo zajistěte použití argumentu **-u** pro AMQSCLM.

Potenciální vysvětlení: U žádného správce front nejsou k dispozici žádné aktivní fronty. Zprávy jsou rovnoměrně rozloženy do všech neaktivních front, dokud se fronta nestane aktivní.

Akce: Zkontrolujte stav front ve všech správcích front.

Potenciální vysvětlení: Zprávy se umístí do jiného správce front v klastru do té, která vlastní neaktivní frontu, a aktualizovaná hodnota **CLWLPRTY** 0 není šířena do správce front pro uvedení aplikace.

Akce: Zkontrolujte, zda jsou spuštěny kanály klastru mezi monitorovaným správcem front a úplným správcem front úložiště. Zkontrolujte, zda jsou spuštěny kanály mezi umístěním správce front a správce front úplného úložiště. Zkontrolujte protokoly chyb monitorovaných, umísťovačů a správců front úplného úložiště.

Potenciální vysvětlení: Instance vzdálených front jsou aktivní (CLWLPRTY=1), ale zprávy nelze směrovat na tyto instance fronty, protože odesílací kanál klastru z lokálního správce front není spuštěn.

Akce: Zkontrolujte stav odesílacích kanálů klastru z lokálního správce front do vzdáleného správce front nebo správců s aktivní instancí fronty.

Scénář: AMQSCLM nepřevádí zprávy z neaktivní fronty

Potenciální vysvětlení: Přenos zpráv není povolen (**-t**).

Akce: Ujistěte se, že přenos zpráv je povolen (**-t**).

Potenciální vysvětlení: Fronta se nenachází v seznamu front, které mají být monitorovány.

Akce: Zkontrolujte hodnoty parametrů **-q** a **-f** .

Potenciální vysvětlení: AMQSCLM není spuštěn pro tento správce front nebo pro jiné správce front v klastru, kteří vlastní instance stejné fronty.

Akce: Spusťte AMQSCLM.

Potenciální vysvětlení: Fronta má **CLWLUSEQ**=LOCAL nebo **CLWLUSEQ**=QMGR, a argument **-u** není nastaven.

Akce: Nastavte parametr **-u** , nebo změňte konfiguraci fronty nebo správce front na hodnotu ANY.

Potenciální vysvětlení: V klastru nejsou žádné aktivní instance fronty.

Akce: Zkontrolujte instance fronty s hodnotou **CLWLPRTY**1, nebo vyšší.

Potenciální vysvětlení: Instance vzdálených front mají spotřebitele (**IPPROCS**> = 1), ale jsou neaktivní na těchto správcích front (**CLWLPRTY**= 0), protože AMQSCLM nemonitoruje tyto vzdálené instance.

Akce: Ujistěte se, že je v těchto správcích front spuštěn příkaz AMQSCLM a/nebo je fronta v seznamu front, které mají být monitorovány, kontrolou hodnot parametrů **-q** a **-f** .

Potenciální vysvětlení: Instance vzdálených front jsou aktivní (**CLWLPRTY**= 1), ale jsou považovány za neaktivní na lokálním správci front (**CLWLPRTY**= 0). Tato situace nastane, protože aktualizovaná hodnota produktu **CLWLPRTY** není šířena do tohoto správce front.

Akce: Ujistěte se, že jsou vzdálení správci front připojeni k alespoň jednomu správci front úložiště v klastru. Ujistěte se, že správci front úplného úložiště pracují správně. Zkontrolujte, zda jsou spuštěny kanály mezi správci front úplného úložiště a monitorovanými správci front.

Potenciální vysvětlení: Zprávy nejsou potvrzeny, proto nemohou být vyhledatelné.

Akce: Zkontrolujte, zda odesílající aplikace funguje správně.

Potenciální vysvětlení: AMQSCLM nemá přístup k lokální frontě, kde jsou zprávy zařazeny do fronty.

Akce: Tento scénář může být způsoben tím, že AMQSCLM není spuštěn jako uživatel s dostatečnou autorizací pro přístup k frontě.

Potenciální vysvětlení: Příkazový server správce front není spuštěn.

Akce: Spusťte příkazový server správce front.

Možné vysvětlení: AMQSCLM rozpoznal chybu.

Akce: Zkontrolujte, zda soubory sestav nejsou chyby.

Potenciální vysvětlení: Instance vzdálených front jsou aktivní (CLWLPRTY=1), ale zprávy nelze přenést na tyto instance fronty, protože odesílací kanál klastru z lokálního správce front není spuštěn. K této akci je často připojeno varování CLM0030W v protokolu sestavy amqsclm.

Akce: Zkontrolujte stav odesílacích kanálů klastru z lokálního správce front do vzdáleného správce front nebo správců s aktivní instancí fronty.

Ukázkový program pro vyhledávání koncového bodu připojení (CEPL)

Ukázka Vyhledávání koncového bodu připojení produktu IBM WebSphere MQ poskytuje jednoduchý, ale účinný modul uživatelské procedury, který nabízí uživatelům produktu WebSphere MQ způsob, jak načíst definice připojení z úložiště LDAP, jako je například produkt Tivoli Directory Server.

Musí být nainstalován produkt Tivoli Directory Server v6.3 Client, aby bylo možné používat CEPL.

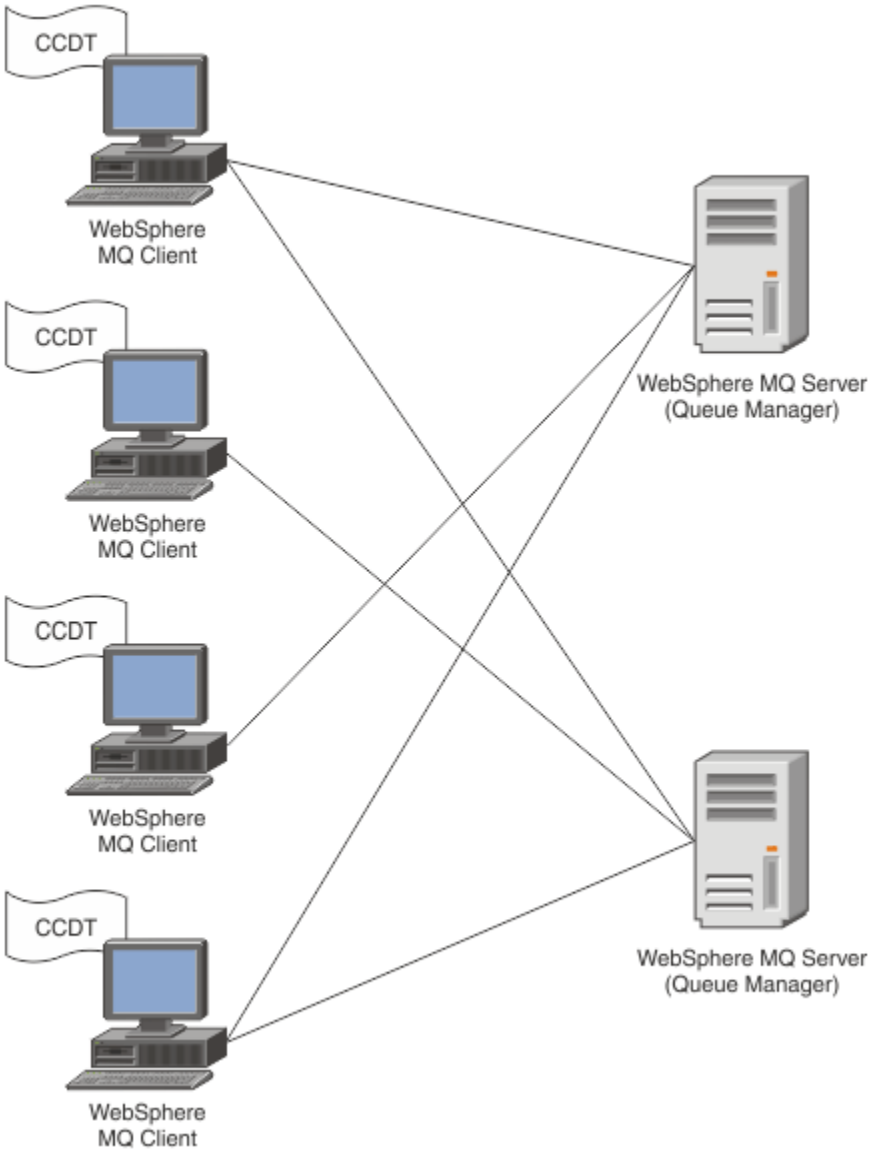
K použití této ukázky je nezbytné funkční znalosti administrace produktu WebSphere MQ na podporovaných platformách.

Úvod

Nakonfigurujte globální úložiště, například adresář LDAP (Lightweight Directory Access Protocol), chcete-li uložit definice připojení klienta pro podporu údržby a administrace.

Pomocí aplikace klienta IBM WebSphere MQ lze navázat spojení se správcem front prostřednictvím tabulky CCDT (Client Connection Definition Table).

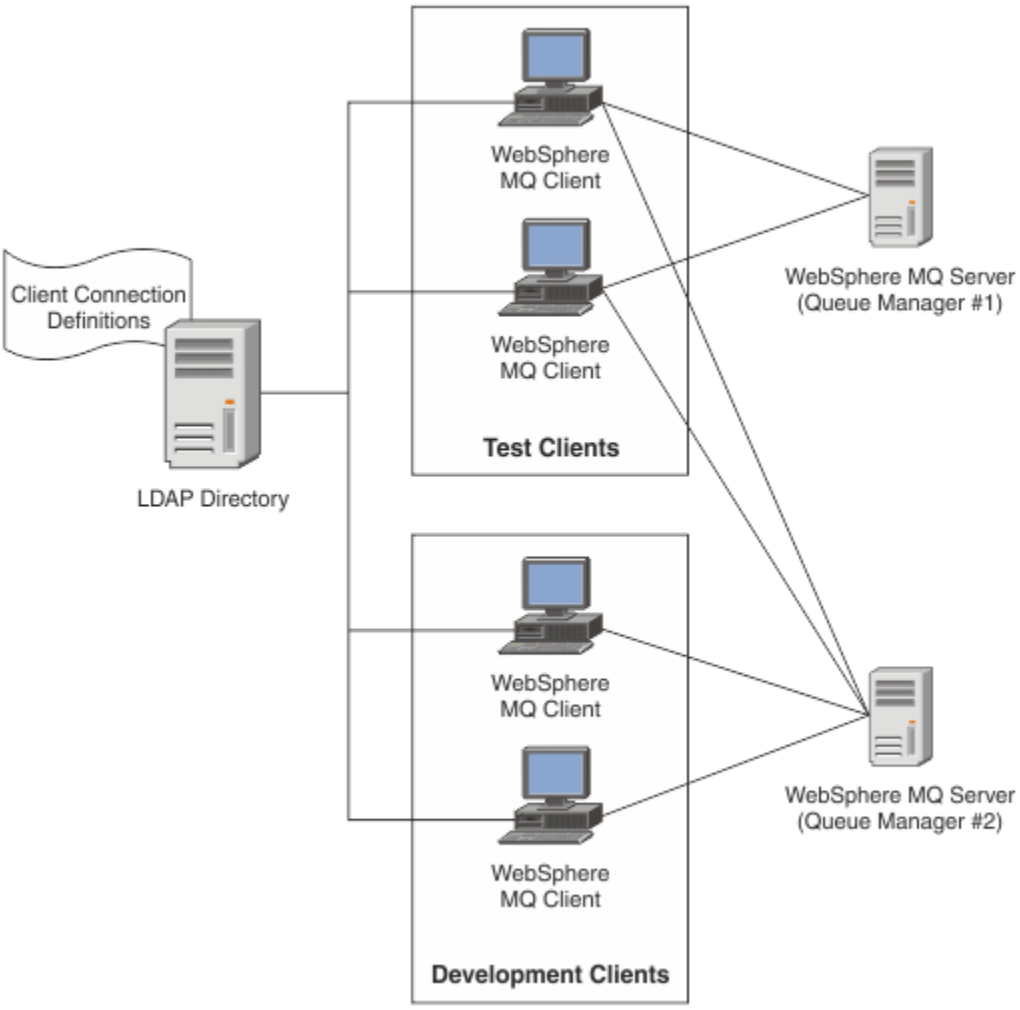
Nástroje CCDT se vytvářejí prostřednictvím standardního rozhraní pro administraci produktu WebSphere MQ MQSC. Uživatel musí být připojen ke správci front, aby bylo možné vytvořit definice připojení klienta, i když data obsažená v definici nejsou omezena na správce front. Generovaný soubor CCDT musí být ručně distribuován mezi klientskými počítači a aplikacemi.



Soubor CCDT musí být distribuován na každého klienta WebSphere MQ . V případě, že tisíce klientů mohou existovat buď lokálně, nebo globálně, brzy by bylo obtížné udržovat a spravovat. Je zapotřebí flexibilnějšího přístupu, který pomáhá zajistit, aby každý klient měl k dispozici správné definice klienta.

Jednou z takových přístupů je uložit definice připojení klienta do globálního úložiště, jako je například adresář LDAP (Lightweight Directory Access Protocol). Adresář LDAP může také poskytovat další funkce zabezpečení, indexace a vyhledávání, čímž umožňuje každému klientovi přístup pouze k těm definicím připojení, které se jich týkají.

Adresář LDAP lze nakonfigurovat tak, aby byly k dispozici pouze určité definice pro určité skupiny uživatelů. Testovací klienti mohou například přistupovat k oběma správci front #1 i k produktu #2, zatímco klienti Development Clients mohou přistupovat pouze k produktu Queue Manager #2 .



Výstupní modul může vyhledat úložiště LDAP, například produkt IBM Tivoli Directory Server, aby načetl definice kanálů. Pomocí těchto definic připojení může klientská aplikace produktu WebSphere MQ navázat spojení se správcem front.

Výstupní modul je předpřipojovacím uživatelským modulem, který umožňuje získat definici kanálu během volání MQCONN/MQCONNX z úložiště LDAP.

Výstupní modul a schéma mohou být implementovány pomocí:

- Zákazníci, kteří již vybudovali základ dovednosti s využitím existující technologie založené na souborech CCDT a chtějí snížit náklady na administraci a distribuci.
- Existující zákazníci, kteří již používají svou vlastní proslušnou technologii pro distribuci definic připojení klientů.

- Noví nebo stávající zákazníci, kteří momentálně nepoužívají žádný typ řešení připojení klienta a chtějí využívat funkce nabízené produktem IBM WebSphere MQ.
- Noví nebo stávající zákazníci, kteří chtějí přímo používat nebo ladit svůj model systému zpráv vložený s libovolnou aktuální obchodní architekturou LDAP.

Podporovaná prostředí

Před spuštěním ukázky vyhledání koncového bodu připojení ověřte, že máte podporovaný operační systém a odpovídající software.

Ukázkový program pro IBM WebSphere MQ Vyhledávání koncového bodu připojení vyžaduje následující software:

- IBM WebSphere MQ V7.0nebo novější
- Klient Tivoli Directory Server V6.3 nebo novější

Podporované operační systémy:

1. Windows (XP/2003/2008)
2. Solaris (SPARC a x86-64)
3. AIX
4. Linux
   - RHEL v4 a v5 na systému System p
   - SUSE v9 a v10 na System p
   - RHEL v4 a v5 System x32 bitů a x64 bitů
   - SUSE v9 a v10 System x32 bitů a x64 bitů
5. HP IA64.

Poznámka: Ukázkový program není k dispozici pro platformy z/OS, i/5a HP PARISC.

Instalace a konfigurace

Instalace a konfigurace výstupního modulu a schématu koncového bodu připojení.

Instalace výstupního modulu

Během instalace produktu WebSphere MQje modul uživatelské procedury instalován pod `tools/samples/c/preconnexit/bin`. Pro 32bitové platformy musí být výstupní modul zkopírován do produktu `exit/<install name>/` dříve, než jej lze použít. Na 64bitových platformách musí být výstupní modul zkopírován do adresáře exit64/<installation >/před tím, než jej lze použít.

Instalace schématu koncového bodu připojení

Uživatelská procedura používá schéma koncového bodu připojení, *ibm-amq.schema* . Před použitím uživatelské procedury musí být soubor schématu importován do libovolného serveru LDAP. Po importu schématu musí být přidány hodnoty pro atributy.

Zde je příklad pro import schématu koncového bodu připojení. Příklad předpokládá, že se používá produkt IBM Tivoli Directory Server (ITDS).

- Ujistěte se, že IBM Tivoli Directory Server běží, pak zkopírujte nebo FTP soubor *ibm-amq.schema* na server ITDS.
- Na serveru ITDS zadejte následující příkaz, který nainstaluje schéma do úložiště ITDS, kde ID LDAP a heslo LDAP jsou kořenové DN a heslo pro server LDAP:

  ldapadd -D "LDAP ID" -w "LDAP heslo" -f ibm-amq.schema
- V příkazovém okně zadejte následující příkaz nebo použijte nástroj třetí strany k procházení schématu pro ověření:

  ldapsearch objectclass=ibm-amqClientConnection

Další podrobnosti o importu souboru schématu najdete v dokumentaci k serveru LDAP.

Konfigurace

Do konfiguračního souboru klienta s názvem *mqclient.ini* musí být přidána nová sekce s názvem **PreConnect** . Sekce PreConnect obsahuje následující klíčová slova:

Modul : Název modulu, který obsahuje kód ukončení rozhraní API. Pokud toto pole obsahuje úplnou cestu k modulu, bude použita stejně jako složka *exit* nebo složka *exit64* v instalaci produktu WebSphere MQ .

Funkce : Název funkčního vstupního bodu do knihovny, která obsahuje výstupní kód PreConnect . Definice funkce dodržuje prototyp MQ_PRECONNECT_EXIT.

Data : URI úložiště LDAP obsahujícího definice kanálů.

Následující úsek kódu je příkladem změn nezbytných pro soubor *mqclient.ini* .

```
PreConnect:
  Module=amqlcelp
  Function=PreConnectExit
  Data=ldap://myLDAPServer.com:389/cn=wmq,ou=ibm,ou=com
  Sequence=1
```

Přehled o ukončení a schématu

Syntaxe a parametry použité k vytvoření připojení ke správci front.

WebSphere MQ v7.5 definuje následující syntaxi pro vstupní bod v modulu uživatelské procedury.

```
void MQENTRY MQ_PRECONNECT_EXIT ( PMQNXP  pExitParms
                                , PMQCHAR pQMgrName
                                , PPMQCNO ppConnectOpts
                                , PMQLONG pCompCode
                                , PMQLONG pReason)
```

Během provádění volání MQCONN/X klient WebSphere MQ C Client načte výstupní modul obsahující implementaci syntaxe funkce. Pak vyvolá funkci ukončení, aby načetla definice kanálu. Načtené definice kanálu se poté používají k navázání spojení se správcem front.

Parametry

pExitParms

Typ: PMQNXP I/O

Struktura konfiguračního parametru PreConnection . Tato struktura je přidělována a udržována volajícím pro ukončení.

```
struct tagMQNXP
{
  MQCHAR4    StrucId;        /* Structure identifier */
  MQLONG    Version;        /* Structure version number */
  MQLONG    ExitId;             /* Type of exit */
  MQLONG    ExitReason;         /* Reason for invoking exit */
  MQLONG    ExitResponse;       /* Response from exit */
  MQLONG    ExitResponse2;      /* Secondary response from exit */
  MQLONG    Feedback;           /* Feedback code (reserved) */
  MQLONG    ExitDataLength;        /* Exit data length */
  PMQCHAR      pExitDataPtr;           /* Exit data */
  MQPTR        pExitUserAreaPtr;       /* Exit user area */
  PMQCD *    ppMQCDArrayPtr;        /* Array of pointers to MQCDs */
  MQLONG    MQCDArrayCount;        /* Number of entries found */
  MQLONG    MaxMQCDVersion;        /* Maximum MQCD version */
};
```

pQMgr

Typ: PMQCHAR vstupní/výstupní

Název správce front. Na vstupu je tento parametr řetězcem filtru dodaným do volání rozhraní API MQCONN prostřednictvím parametru **QMgrName** . Toto pole může být prázdné, explicitní nebo obsahovat určité zástupné znaky. Pole je změněno uživatelskou procedurou. Při volání procedury MQXR_TERM je parametr nastaven na hodnotu Null.

ppConnectOtty

Typ: ppConnectOpts vstup/výstup

Volby, které řídí akci MQCONNX. Jedná se o ukazatel na strukturu voleb připojení MQCNO, která řídí akci volání rozhraní API MQCONN. Při volání procedury MQXR_TERM je parametr nastaven na hodnotu Null. Klient MQI vždy poskytuje strukturu MQCNO pro ukončení i v případě, že ji aplikace původně neposkytovala. Pokud aplikace poskytuje strukturu MQCNO, klient vytvoří duplikát a předá jej do uživatelské procedury, kde je upravena. Klient si zachová vlastnictví objektu MQCNO. MQCD, na které odkazuje objekt MQCNO, má přednost před jakoukoli definicí připojení poskytnutou prostřednictvím pole. Klient používá strukturu MQCNO k připojení ke správci front a ostatní jsou ignorovány.

pCompKód

Typ: PMQLONG vstupní/výstupní

Kód dokončení. Ukazatel na MQLONG, který přijímá kód dokončení ukončení. Musí se jednat o jednu z následujících hodnot:

MQCC_OK-Úspěšné dokončení

MQCC_WARNING-Varování (částečné dokončení)

MQCC_FAILED-Volání se nezdařilo

pReason

Typ: PMQLONG vstupní/výstupní

Kód určující kvalifikaci pCompCode. Ukazatel na hodnotu MQLONG, která přijímá kód příčiny ukončení. Je-li kód dokončení MQCC_OK, jediná platná hodnota je:

MQRC_NONE-(0, x '000') Chybí důvod k vytvoření sestavy.

Je-li kód dokončení MQCC_FAILED nebo MQCC_WARNING, může funkce uživatelské procedury nastavit pole s kódem příčiny na jakoukoli platnou hodnotu MQRC_ *.

Informace o kontextu služby LDAP produktu MQ

Uživatelská procedura používá následující strukturu dat pro kontextové informace.

MQNLDAPCTX

Struktura MQNLDAPCTX má následující prototyp C.

```
typedef struct tagMQNLDAPCTX MQNLDAPCTX;
typedef MQNLDAPCTX MQPOINTER PMQNLDAPCTX;

struct tagMQNLDAPCTX
{
  MQCHAR4    StrucId;        /* Structure identifier */
  MQLONG    Version;        /* Structure version number */
  LDAP *    objectDirectory;    /* LDAP Instance */
  MQLONG       ldapVersion;        /* Which LDAP version to use? */
  MQLONG       port;            /* Port number for LDAP server*/
  MQLONG       sizeLimit;        /* Size limit */
  MQBOOL    ssl;            /* SSL enabled? */
  MQCHAR *    host;            /* Hostname of LDAP server */
  MQCHAR *    password;        /* Password of LDAP server */
  MQCHAR *    searchFilter;        /* LDAP search filter */
  MQCHAR *    baseDN;            /* Base Distinguished Name */
  MQCHAR *    charSet;        /* Character set */
};
```

Ukázkový kód pro sestavení uživatelské procedury pro vyhledání koncového bodu připojení

Úseky kódu pro kompilaci zdroje na systému Windows a distribuovaných platformách.

Kompilace zdroje

Zdroj můžete kompilovat s libovolými knihovnami klienta LDAP, například knihovnami klienta IBM Tivoli Directory Server 6.3 . Tato dokumentace předpokládá, že používáte knihovny klienta produktu Tivoli Directory Server 6.3 .

Poznámka: Knihovna uživatelské procedury před připojením byla testována s následujícími servery LDAP:

- IBM Tivoli Directory Server V6.3
- Novell eDirectory V8.2

Následující úseky kódu popisují, jak zkompilovat uživatelské procedury na systému Windowsa jiné distribuované platformy:

Kompilování uživatelské procedury na platformě Windows

Při kompilaci zdroje ukončení v systému Windowsmůžete 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: Při kompilaci knihoven klienta IBM Tivoli Directory Server 6.3 s produktem Microsoft Visual Studio 2005 nebo nad kompilátorem byste mohli zobrazit varování, pokud používáte knihovny klienta IBM Tivoli Directory Server 6.3 , které jsou kompilovány s kompilátorem Microsoft Visual Studio 2003.

Kompilace uživatelské procedury na jiných distribuovaných platformách

Následující úsek kódu můžete použít ke kompilaci výstupního zdroje na jiných distribuovaných platformách, např. Linux. Některé volby kompilátoru se mohou lišit i na jiných distribuovaných platformách.

```
#Make file to build exit
CC=gcc

MQML=/opt/mqm/lib
MQMI=/opt/mqm/inc
TDSI=/opt/ibm/ldap/V6.3/include
XFLAG=-m32

TDSL=/opt/ibm/ldap/V6.3/lib
```

IBM Tivoli Directory Server je dodáván jak statických, tak i dynamicky propojovacím knihovnami, ale lze použít pouze jeden formulář knihoven. 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)
```

Vyvolání výstupního modulu

Modul uživatelské procedury PreConnect lze vyvolat se třemi různými kódy příčin. Tato sekce popisuje každý důvod ukončení ve větší hloubce.

MQXR_INIT

Ukončení je vyvoláno s kódem příčiny MQXR_INIT pro inicializaci a navázání spojení se serverem LDAP.

Před voláním funkce *MQXR_INIT* by pole *pExitDataPtr* struktury MQNXP bylo naplněno atributem Data ze sekce PreConnect v souboru *mqclient.ini* (tj. LDAP).

Adresa URL protokolu LDAP se skládá alespoň z protokolu, názvu hostitele, čísla portu a základního rozlišujícího názvu pro hledání. Uživatelská procedura analyzuje adresu URL protokolu LDAP obsaženou v poli *pExitDataPtr* , přidělí strukturu kontextu vyhledávání LDAP MQNLDAPCTX a naplní ji odpovídajícím způsobem. Adresa této struktury je uložena v poli *pExitUserAreaPtr* . Selhání při správné analýze výsledků adresy URL protokolu LDAP se nezdařilo, došlo k chybě MQCC_FAILED.

V tomto okamžiku se uživatelská procedura připojí k serveru LDAP pomocí parametrů MQNLDAPCTX a připojí se k ní. Výsledné popisovače rozhraní LDAP API jsou také uloženy v této struktuře.

PŘIPOJENÍ MQXR_PRECONNECT
Výstupní modul je vyvolán s kódem příčiny MQXR_PRECONNECT pro načítání definic kanálů ze serveru LDAP.

Ukončení prohledává server LDAP pro definice kanálu odpovídající danému filtru. Pokud parametr *QMgrName* obsahuje specifický název správce front, pak vyhledávání vrátí všechny definice kanálů, jejichž hodnota atributu LDAP *ibm-amqQueueManagerName* odpovídá názvu daného správce front.

Je-li parametr *QMgrName* '*' nebo ' ' (mezera), pak hledání vrátí všechny definice kanálu, jejichž atribut koncového bodu připojení *ibm-amqIsClientDefault* je nastaven na hodnotu true.

Po úspěšném hledání procedura připraví jeden nebo pole definic MQCD a vrátí se zpět volajícímu.

MQXR_TERM
Ukončení je vyvoláno s tímto kódem příčiny, když se má ukončit ukončení uživatelské procedury. Během tohoto ukončení se ukončení odpojí od serveru LDAP, uvolní veškerou přidělenou a udržovanou paměť uživatelskou procedurou. To bude zahrnovat strukturu *MQNLDAPCTX* , pole ukazatele a každý odkaz MQCD, na který se odkazuje. Všechna ostatní pole jsou nastavena na výchozí hodnoty. Parametry uživatelské procedury *pQMgrName* a *ppConnectOpts* se během operace *MQXR_TERM* nepoužívají a mohou mít hodnotu Null.

Schémata LDAP
Data připojení klienta jsou uložena v globálním úložišti, označeném jako adresář LDAP (Lightweight Directory Access Protocol). Klient WebSphere MQ používá adresář LDAP k získání definic připojení. Struktura definic připojení klienta WebSphere MQ v rámci adresáře LDAP je známá jako schéma LDAP. Schéma LDAP je kolekce definic typů atributů, definic tříd objektů a dalších informací, které server používá k určení, zda se shoda filtru nebo atributu hodnoty shoduje s atributy záznamu, a zda se má povolit, přidat a upravit operace.

Ukládání dat do adresáře LDAP

Definice připojení klienta jsou umístěny pod specifickou větví v adresářovém stromu známém jako přípojný bod. Podobně jako všechny ostatní uzly v rámci adresáře LDAP má připojovací bod k sobě přidružený rozlišující název (DN). Tento uzel můžete použít jako výchozí bod pro všechny dotazy, které vytvoříte v adresáři. Použijte filtrování při dotazu na adresář LDAP k vrácení podmnožiny definic připojení klienta. Přístup k podstromům můžete omezit na základě oprávnění udělených v jiných částech adresářového stromu-například pro uživatele, oddělení nebo skupiny.

Definování vlastních atributů a tříd
Uložte definici kanálu klienta úpravou schématu LDAP. Všechny definice dat LDAP vyžadují objekty a atributy. Objekty a atributy jsou identifikovány identifikátorem OID (Object Identifier), který jednoznačně identifikuje objekt nebo atribut. Všechny třídy v rámci schématu LDAP dědí buď přímo, nebo nepřímo od horního objektu. Objekt definice kanálu klienta obsahuje atributy horního objektu. Všechny definice dat LDAP vyžadují objekty a atributy:

- Definice objektů jsou kolekce atributů LDAP.
- Atributy jsou datové typy LDAP.

Popis jednotlivých atributů a jejich mapování na běžné vlastnosti produktu WebSphere MQ je popsán v tématu Atributy LDAP .

Atributy LDAP

Definované atributy LDAP jsou specifické pro produkt WebSphere MQ a jsou mapovány přímo na vlastnosti připojení klienta.

Atributy řetězce adresáře kanálu klienta WebSphere MQ

Atributy znakového řetězce s jejich mapováním na vlastnosti produktu WebSphere MQ jsou uvedeny v následující tabulce. Atributy mohou obsahovat hodnoty directoryString (UTF-8 encoded Unicode, to je proměnná byte encoding systému, která obsahuje syntaxi IA5/ASCII jako dílčí sadu). Syntaxe je uvedena jeho identifikačním číslem objektu (OID).

Tabulka 21. Atributy řetězce adresáře kanálu klienta produktu WebSphere MQ

atribut LDAP	Popis	Vlastnost produktu WebSphere MQ
CN	Obecný název sestávající z názvu kanálu a názvu definujícího správce front.	
ibm-amqChannelNázev	Název definice kanálu.	CHANNEL
ibm-amqConnectionNázev	Identifikátor komunikačního připojení.	CONNAME
ibm-amqDescription	Popis kanálu.	DESCR
ibm-amqLocal	Lokální komunikační adresa kanálu.	LOCLADDR
ibm-amqModeNázev	s názvem režimu bThe LU 6.2 .	MODENAME
ibm-amqPassword	Heslo, které lze použít.	PASSWORD
ibm-amqQueueManagerName	Název správce front nebo skupiny správců front, ke které může klientská aplikace produktu WebSphere MQ požadovat připojení.	QMNAME
ibm-amqSecurityExitUserData	Uživatelská data, která jsou předána uživatelské proceduře pro zabezpečení zprávy.	SCYDATA
ibm-amqSecurityExitName	Název ukončovacího programu, který má být spuštěn uživatelskou procedurou zabezpečení kanálu.	SCYEXIT
ibm-amqSslCipherSpec	Jedna CipherSpec pro připojení SSL.	SSLCIPH
ibm-amqSslPeerName	Zkontroluje rozlišující název (DN) certifikátu od partnerského správce front nebo klienta na druhém konci kanálu WebSphere MQ .	SSLPEER
ibm-amqTransactionProgramName	Název transakčního programu.	TPNAME
ibm-amqUserID	Jméno uživatele, které má být použito agentem MCA při pokusu o zahájení zabezpečené relace SNA se vzdáleným agentem MCA.	USERID

Celočíselné atributy připojení klienta WebSphere MQ

Atributy s předdefinovanými hodnotami (například výčtový typ) jsou uloženy jako standardní celá čísla. Tyto hodnoty jsou uloženy v adresáři LDAP jako celočíselné hodnoty, a ne pomocí přidruženého názvu konstanty.

Tabulka 22. Celočíselné atributy adresáře kanálu klienta produktu WebSphere MQ		
atribut LDAP	**Popis**	**Vlastnost produktu WebSphere MQ**
Afinita ibm-amqConnection	Určuje, zda klientské aplikace, které se připojují vícekrát přes stejné jméno správce front, používají stejný kanál klienta.	AFFINITY
ibm-amqClientChannelWeight	Váhový faktor, který ovlivňuje použitou definici kanálu připojení klienta.	CLNTWGHT
ibm-amqHeartBeatInterval	Přibližný čas mezi toky synchronizačních signálů předávanými z agenta kanálu zpráv v případě, kdy se v přenosové frontě nenacházejí žádné zprávy.	HBINT
ibm-amqKeepAliveInterval	Hodnota časového limitu pro kanál.	KAINT
ibm-amqMaximumMessageLength	Maximální délka zprávy, kterou lze přenést v kanálu.	MAXMSGL
ibm-amqSharingKonverzace	Maximální počet konverzací, které sdílejí každou instanci kanálu TCP/IP.	SHARECNV
ibm-amqTransportTyp	Typ transportu, který má být použit.	TRPTYPE

Logický atribut kanálu klienta WebSphere MQ

Tento logický atribut není mapován na žádnou vlastnost produktu WebSphere MQ . Syntaxe tohoto atributu označuje logickou hodnotu.

Tabulka 23. Logický atribut kanálu klienta WebSphere MQ	
atribut LDAP	**Popis**
ibm-amqIsClientDefault	Tento logický atribut je definován pro vyřešení problému při hledání položek, jejichž atribut ibm-amqQueueManagerName nebyl definován.

Atributy seznamu kanálů klienta produktu WebSphere MQ

Vlastnosti produktu WebSphere MQ se ukládají jako atribut seznamu s jedinou hodnotou, oddělený čárkami v adresáři LDAP. Atributy se definují stejným způsobem jako ostatní atributy řetězce adresáře. Atributy seznamu spolu s jejich mapováním na vlastnosti produktu WebSphere MQ jsou popsány v následující tabulce.

Tabulka 24. Atributy seznamu kanálů klienta produktu WebSphere MQ		
atribut LDAP	**Popis**	**Vlastnost produktu WebSphere MQ**
ibm-amqHeaderKomprese	Seznam metod komprese dat záhlaví podporovaných kanálem.	COMPHDR
ibm-amqMessageCompression	Seznam technik komprese dat zpráv podporovaných kanálem.	COMPMSG
ibm-amqSendExitUserData	Uživatelská data, která jsou předána uživatelské proceduře pro odeslání zprávy.	SENDDATA
ibm-amqSendExitUserNázev	Název ukončovacího programu, který má být spuštěn uživatelskou procedurou odeslání kanálu.	SENDEXIT

Tabulka 24. Atributy seznamu kanálů klienta produktu WebSphere MQ (pokračování)		
atribut LDAP	**Popis**	**Vlastnost produktu WebSphere MQ**
ibm-amqReceiveExitUserData	Uživatelská data, která jsou předána uživatelské proceduře pro přijetí zprávy.	RCVDATA
ibm-amqReceiveExitName	Název uživatelského ukončovacího programu, který má být spuštěn uživatelskou procedurou kanálu pro přijetí zprávy.	RCVEXIT

Obecný název
Obecný název (CN) se skládá z názvu kanálu a definování názvu správce front.

Jedná se o předexistující atribut.

Formát KN je:

```
CN=CHANNEL_NAME(DEFINING_Q_MGR_NAME)
```

Příklad:

```
CN=TC1(QM_T1)
```

Pro tento atribut můžete zadat pouze jednu hodnotu.

Tento atribut je řetězcový atribut a hodnoty nerozlišují velikost písmen. Shoda podřetězce je ignorována. Porovnávání podřetězce je porovnávací pravidlo použité v podschématu, které určuje chování atributu ve filtru vyhledávání, s použitím podřetězce (například CN=jim *, kde CN je atribut) a obsahuje jednu nebo více zástupných znaků.

Název ibm-amqChannel
Tento atribut určuje název definice kanálu.

Tento atribut má jedinou řetězcovou hodnotu s maximálně 20 znaky, které nejsou citlivé na velikost písmen. Není to již existující atribut.

Shoda podřetězce je ignorována. Porovnávání podřetězce je porovnávací pravidlo použité v dílčím schématu, které určuje chování atributu ve filtru vyhledávání, pomocí podřetězce a obsahuje jednu nebo více zástupných znaků.

ibm-amqDescription
Tento atribut LDAP poskytuje popis kanálu.

Tento atribut má jedinou řetězcovou hodnotu s maximálně 64 bajty, které nejsou citlivé na velikost písmen. Není to již existující atribut.

Shoda podřetězce je ignorována. Porovnávání podřetězců je odpovídající pravidlo použité v dílčím schématu, které určuje chování atributu ve filtru vyhledávání.

Název ibm-amqConnection
Tento atribut LDAP je identifikátor komunikačního připojení. Určuje konkrétní komunikační spojení, které má tento kanál používat.

Tento atribut má jedinou řetězcovou hodnotu s maximálně 264 znaky, které nejsou citlivé na velikost písmen. Není to již existující atribut.

Shoda podřetězce je ignorována. Porovnávání podřetězců je odpovídající pravidlo použité v dílčím schématu, které určuje chování atributu ve filtru vyhledávání.

Adresa ibm-amqLocal
Tento atribut určuje adresu lokální komunikace pro kanál.

Tento atribut má jedinou řetězcovou hodnotu s maximálně 48 znaky, které nejsou citlivé na velikost písmen. Není to již existující atribut.

Shoda podřetězce je ignorována. Porovnávání podřetězců je odpovídající pravidlo použité v dílčím schématu, které určuje chování atributu ve filtru vyhledávání.

Název ibm-amqMode
Tento atribut je určen pro použití s připojeními LU 6.2. Poskytuje další definici charakteristik relace připojení, když se provádí alokace komunikační relace.

Tento atribut má jedinou řetězcovou hodnotu přesně 8 znaků, které nejsou citlivé na velikost písmen. Není to již existující atribut.

Shoda podřetězce je ignorována. Porovnávání podřetězců je odpovídající pravidlo použité v dílčím schématu, které určuje chování atributu ve filtru vyhledávání.

ibm-amqPassword
Tento atribut LDAP určuje heslo, které může být použito agentem MCA při pokusu o zahájení zabezpečené relace LU 6.2 se vzdáleným agentem MCA.

Tento atribut má jedinou celočíselnou hodnotu o maximální délce 12 číslic. Není to již existující atribut.

ibm-amqQueueManagerName
Tento atribut určuje název správce front nebo skupiny správců front, ke které může klientská aplikace produktu WebSphere MQ požadovat připojení.

Tento atribut má jedinou řetězcovou hodnotu s maximálně 48 znaky, které nejsou citlivé na velikost písmen. Není to již existující atribut.

Shoda podřetězce je ignorována. Porovnávání podřetězců je odpovídající pravidlo použité v dílčím schématu, které určuje chování atributu ve filtru vyhledávání.

Data ibm-amqSecurityExitUser
Tento atribut LDAP určuje uživatelská data, která jsou předána uživatelské proceduře pro zabezpečení zprávy.

Tento atribut má jedinou řetězcovou hodnotu s maximálně 999 znaky, které nejsou citlivé na velikost písmen. Není to již existující atribut.

Shoda podřetězce je ignorována. Porovnávání podřetězců je odpovídající pravidlo použité v dílčím schématu, které určuje chování atributu ve filtru vyhledávání.

ibm-amqSecurityExitName
Tento atribut LDAP uvádí název ukončovacího programu, který má být spuštěn uživatelskou procedurou zabezpečení kanálu.

Ponechte prázdné, není-li v platnosti žádná uživatelská procedura zabezpečení kanálu.

Tento atribut má jedinou řetězcovou hodnotu s maximálně 999 znaky, které nejsou citlivé na velikost písmen. Tento atribut není předukončující.

Shoda podřetězce je ignorována. Porovnávání podřetězců je odpovídající pravidlo použité v dílčím schématu, které určuje chování atributu ve filtru vyhledávání.

ibm-amqSslCipherSpec
Tento atribut LDAP určuje jednu položku CipherSpec pro připojení SSL.

Tento atribut má jedinou řetězcovou hodnotu o maximální délce 32 znaků, které nejsou citlivé na velikost písmen. Není to již existující atribut.

Shoda podřetězce je ignorována. Porovnávání podřetězců je odpovídající pravidlo použité v dílčím schématu, které určuje chování atributu ve filtru vyhledávání.

ibm-amqSslPeerName
Tento atribut LDAP slouží ke kontrole rozlišujícího názvu (DN) certifikátu od partnerského správce front nebo klienta na druhém konci kanálu produktu WebSphere MQ .

Tento atribut LDAP má jedinou řetězcovou hodnotu s maximálně 1024 bajty, které nejsou citlivé na velikost písmen. Není to již existující.

Shoda podřetězce je ignorována. Porovnávání podřetězců je odpovídající pravidlo použité v dílčím schématu, které určuje chování atributu ve filtru vyhledávání.

ibm-amqTransactionProgramName
Tento atribut LDAP uvádí název transakčního programu. Používá se pro připojení LU 6.2 .

Tento atribut má jedinou řetězcovou hodnotu o maximální délce 64 znaků, která nerozlišuje velikost písmen. Není to již existující.

Shoda podřetězce je ignorována. Porovnávání podřetězců je odpovídající pravidlo použité v dílčím schématu, které určuje chování atributu ve filtru vyhledávání.

ID ibm-amqUser
Tento atribut LDAP určuje jméno uživatele, které má být použito agentem MCA při pokusu o zahájení zabezpečené relace SNA se vzdáleným agentem MCA.

Tento atribut má jedinou řetězcovou hodnotu přesně 12 znaků, které nejsou citlivé na velikost písmen. Není to již existující atribut.

Shoda podřetězce je ignorována. Porovnávání podřetězců je odpovídající pravidlo použité v dílčím schématu, které určuje chování atributu ve filtru vyhledávání.

Afinita ibm-amqConnection
Tento atribut LDAP určuje, zda klientské aplikace, které se připojují vícekrát pomocí stejného názvu správce front, používají stejný kanál klienta.

Tento atribut má jedinou celočíselnou hodnotu. Není to již existující atribut.

ibm-amqClientChannelWeight
Tento atribut LDAP určuje váhu ovlivňující definici kanálu připojení klienta, která se má použít.

Váhový atribut kanálu klienta se používá k ovlivnění výběru definic kanálů klienta, je-li k dispozici více než jedna vhodná definice.

Tento atribut má jedinou celočíselnou hodnotu. Není to již existující atribut.

ibm-amqHeartBeatInterval
Tento atribut LDAP uvádí přibližný čas mezi toky synchronizačních signálů, které mají být předány z odesílající sběrnice MCA, když v přenosové frontě nejsou žádné zprávy.

Tento atribut má jedinou celočíselnou hodnotu. Není to již existující atribut. Výchozí hodnota je 1. Výchozí hodnota je nastavena v aktuální operaci proměnné prostředí MQSERVER.

ibm-amqKeepAliveInterval
Tento atribut LDAP se používá k určení hodnoty časového limitu pro kanál.

Hodnota tohoto atributu se předá do komunikačního zásobníku specifikující časování udržení aktivity pro kanál. Tuto hodnotu můžete použít k určení jiné hodnoty udržení aktivity pro každý kanál.

Tento atribut má jedinou celočíselnou hodnotu. Není to již existující atribut.

ibm-amqMaximumMessageLength
Tento atribut LDAP určuje maximální délku zprávy, kterou lze v kanálu přenést.

Výchozí hodnota tohoto atributu je 104857600 tak, jak je za aktuální proměnnou prostředí MQSERVER. Tento atribut má jedinou celočíselnou hodnotu a nejedná se o existující atribut.

Konverzace ibm-amqSharing
Tento atribut LDAP uvádí maximální počet konverzací, které sdílejí každou instanci kanálu TCP/IP.

Tento atribut má jedinou celočíselnou hodnotu. Tento atribut není existující atribut.

Typ ibm-amqTransport
Tento atribut LDAP určuje typ transportu, který má být použit.

Tento atribut má jedinou celočíselnou hodnotu. Není to již existující atribut.

ibm-amqIsClientDefault
Tento logický atribut řeší problém vyhledávání položek, kde atribut ibm-amqQueueManagerName nebyl definován.

Moduly uživatelských procedur před připojením obvykle prohledávají servery LDAP hodnotou atributu ibm-amqQueueManagerName jako vyhledávací kritéria. Takový dotaz by vrátil všechny položky, ve kterých hodnota atributu ibm-amqQueueManagerName odpovídá názvu správce front uvedenému v rámci volání MQCONN/X. Při použití tabulek CCDT (Client Channel Definition CCDT) můžete buď nastavit název správce front na volání MQCONN/X jako prázdný, nebo zadat předponu názvu s hvězdičkou (*). Je-li název správce front prázdný, připojí se klient k výchozímu správci front. Je-li název zadán s hvězdičkou (*) pro správce front, připojí se ke správci front kterýkoli správce front.

Podobně platí, že atribut ibm-amqQueueManagerName v položce může být ponechán nedefinovaný. V tomto případě se očekává, že se klient používající tyto informace koncového bodu může připojit k libovolnému správci front. Položka například obsahuje následující řádky:

```
ibm-amqChannelName = "CHANNEL1"
ibm-amqConnectionName = myhost(1414)
```

V tomto příkladu se klient pokusí o připojení k zadanému správci front spuštěnému v systému `myhost`.

Avšak na serverech LDAP se neprovádí hledání na hodnotě atributu, která nebyla definována. Pokud například položka obsahuje informace o připojení kromě atributu ibm-amqQueueManagerName, výsledky hledání nebudou obsahovat tuto položku. Chcete-li tento problém odstranit, můžete nastavit atribut ibm-amqIsClientDefault. Jedná se o logický atribut a předpokládá se, že má hodnotu FALSE, pokud není definována.

Pro položky, kde parametr ibm-amqQueueManagerName nebyl definován a očekává se, že bude součástí hledání, nastavte atribut ibm-amqIsClientDefault na TRUE. Je-li jako název správce front ve volání MQCONN/X zadán prázdný znak nebo hvězdička (*), uživatelská procedura pro předběžné připojení prohledá server LDAP pro všechny položky, kde hodnota atributu ibm-amqIsClientDefault je nastavena na hodnotu TRUE.

Poznámka: Nenastavujte nebo nadefinujte atribut ibm-amqQueueManagerName , pokud je atribut ibm-amqIsClientDefault nastaven na hodnotu TRUE.

Komprese ibm-amqHeader
Tento atribut LDAP je seznamem technik komprese dat záhlaví, které jsou podporovány kanálem.

Maximální velikost tohoto atributu je 48 znaků. Není to již existující atribut.

Pro tento atribut můžete zadat pouze jednu hodnotu.

Tento atribut seznamu je určen jako řetězce adresáře pomocí formátu odděleného čárkou. Například, hodnoty zadané pro **`ibm-amqHeaderCompression`** jsou `0` , které jsou mapovány na `NONE`. Jakékoli hodnoty, které překročí maximální povolený limit, budou klientem ignorovány. Například ibm-amqHeaderCompression obsahuje maximálně 2 celá čísla v seznamu.

Komprese ibm-amqMessage
Tento atribut LDAP je seznam technik komprese dat zpráv podporovaných kanálem.

Maximální velikost tohoto atributu je 48 znaků. Není to již existující atribut.

Tento atribut nepodporuje více hodnot.

Tento atribut seznamu je určen jako řetězce adresáře pomocí formátu odděleného čárkou. Například, hodnota uvedená pro tento atribut je 1,2,4, která se namapuje na základní pořadí komprese RLE, ZLIBFAST a ZLIBHIGH.

Všechny hodnoty překračující maximální povolený limit je klientem ignorován. Například ibm-amqMessageCompression obsahuje maximálně 16 celých čísel v seznamu.

Data ibm-amqSendExitUserData
Tento atribut LDAP určuje uživatelská data předávaná uživatelské proceduře pro odeslání zprávy.

Tento atribut LDAP má jedinou řetězcovou hodnotu s maximálně 999 znaky, které nejsou citlivé na velikost písmen. Není to již existující atribut.

Shoda podřetězce je ignorována. Porovnávání podřetězců je odpovídající pravidlo použité v dílčím schématu, které určuje chování atributu ve filtru vyhledávání.

Poznámka: ibm-amqSendExitName a **ibm-amqSendExitUserData** je třeba synchronizovat ve dvojicích. Uživatelská data by měla být synchronizována s názvem uživatelské procedury. Je-li tedy jeden zadán, druhý musí být také souměrně specifikován, i když neobsahuje žádná data.

ibm-amqSendExitName
Tento atribut LDAP uvádí název ukončovacího programu, který má být spuštěn uživatelskou procedurou odeslání kanálu.

Tento atribut má jedinou řetězcovou hodnotu s maximálně 999 znaky, které nejsou citlivé na velikost písmen. Není to již existující atribut.

Shoda podřetězce je ignorována. Porovnávání podřetězců je odpovídající pravidlo použité v dílčím schématu, které určuje chování atributu ve filtru vyhledávání.

Poznámka: ibm-amqSendExitName a **ibm-amqSendExitUserData** musí být synchronizované ve dvojicích. Uživatelská data musí být synchronizována s názvem ukončení. Je-li tedy zadán jeden, musí být druhý také symetricky zadán i v případě, že neobsahuje žádná data.

Data ibm-amqReceiveExitUserData
Tento atribut LDAP určuje uživatelská data předávaná uživatelské proceduře pro přijetí zprávy.

Můžete spustit posloupnost uživatelských procedur pro přijetí zprávy. Řetězec uživatelských dat pro řadu uživatelských procedur je oddělen čárkou, mezerami nebo obojím.

Tento atribut má jedinou řetězcovou hodnotu s maximálně 999 znaky, které nejsou citlivé na velikost písmen. Není to již existující atribut.

Shoda podřetězce je ignorována. Porovnávání podřetězců je odpovídající pravidlo použité v dílčím schématu, které určuje chování atributu ve filtru vyhledávání.

Poznámka: ibm-amqReceiveExitName a **ibm-amqReceiveExitUserData** musí být synchronizované ve dvojicích. Uživatelská data musí být synchronizována s názvem ukončení. Je-li tedy zadán jeden, musí být druhý také symetricky zadán i v případě, že neobsahuje žádná data.

ibm-amqReceiveExitName
Tento atribut LDAP uvádí název uživatelského ukončovacího programu, který má být spuštěn uživatelskou procedurou kanálu pro přijetí zprávy.

Tento atribut je seznam názvů programů, které mají být spuštěny v posloupnosti. Ponechte prázdné, pokud není v platnosti žádná uživatelská procedura příjmu kanálu.

Tento atribut má jedinou řetězcovou hodnotu s maximálně 999 znaky, které nejsou citlivé na velikost písmen. Není to již existující atribut.

Shoda podřetězce je ignorována. Porovnávání podřetězců je odpovídající pravidlo použité v dílčím schématu, které určuje chování atributu ve filtru vyhledávání.

Poznámka: ibm-amqReceiveExitName a **ibm-amqReceiveExitUserData** musí být synchronizované ve dvojicích. Uživatelská data musí být synchronizována s názvem ukončení. Je-li tedy zadán, druhý musí být také symetricky zadán, i když neobsahuje žádná data.

Psaní front aplikace

Prostřednictvím těchto informací můžete získat informace o vytváření aplikací pro řazení do front, připojování a odpojování od správce front, publikování a odběru a otevírání a zavírání objektů.

Následující odkazy použijte k vyhledání více informací o psaní aplikací:

Související pojmy

"Koncepty vývoje aplikací" na stránce 7
K zápisu aplikací IBM WebSphere MQ můžete použít volbu procedurálních nebo objektově orientovaných jazyků. Informace o konceptech produktu IBM WebSphere MQ , které jsou užitečné pro vývojáře aplikací, naleznete v odkazech v tomto tématu.

"Rozhodování o tom, jaký programovací jazyk použít" na stránce 75
Tyto informace použijte k vyhledání informací o programovacích jazycích a rámcích podporovaných produktem IBM WebSphere MQa o některých aspektech jejich použití.

"Návrh aplikací produktu IBM WebSphere MQ" na stránce 86
Když se rozhodnete, jak aplikace mohou využívat výhody platforem a prostředí, které máte k dispozici, musíte se rozhodnout, jak budete používat funkce nabízené produktem WebSphere MQ.

"Ukázka programů WebSphere MQ" na stránce 92
Tato kolekce témat slouží k získání informací o ukázkových programech WebSphere MQ na různých platformách.

"Tvorba klientských aplikací" na stránce 337
Co potřebujete vědět, chcete-li zapisovat klientské aplikace do produktu WebSphere MQ.

"Použití webových služeb v produktu WebSphere MQ" na stránce 913
Můžete vyvíjet aplikace produktu IBM WebSphere MQ pro webové služby pomocí přenosu IBM WebSphere MQ pro protokol SOAP nebo mostu produktu IBM WebSphere MQ pro protokol HTTP.

"Sestavení aplikace IBM WebSphere MQ" na stránce 412
Tyto informace použijte k seznámení se s aplikací produktu IBM WebSphere MQ na různých platformách.

"Obsluha chyb programu" na stránce 529
Tyto informace vysvětlují chyby související s voláními MQI MQI buď při volání, nebo při doručení její zprávy do konečného cíle.

Přehled rozhraní fronty zpráv

Zde jsou uvedeny informace o komponentách rozhraní MQI (Message Queue Interface).

Rozhraní fronty zpráv se skládá z následujících prvků:

- *Volání* , jejichž prostřednictvím mohou programy přistupovat ke správci front a k jeho zařízením.
- *Struktury* , které programy používají k předávání dat do správce front a získávání dat z těchto správců front.
- *Elementární datové typy* pro předávání dat do správce front a získávání dat ze správce front

WebSphere MQ for Windows a WebSphere MQ na systémech UNIX and Linux také dodávají:

- Volání, pomocí kterých produkt WebSphere MQ for Windows a WebSphere MQ v systémových programech UNIX and Linux může potvrdit a zazálohovat změny.
- *Zahrnout soubory* , které definují hodnoty konstant dodávaných na těchto platformách.
- *Soubory knihovny* pro propojení vašich aplikací.
- Sada ukázkových programů, které demonstrují, jak používat rozhraní MQI na těchto platformách. Další informace o těchto ukázkách naleznete v tématu "Ukázkové programy pro distribuované platformy" na stránce 93.
- Ukázka zdrojového kódu a spustitelného kódu pro vazby na externí správce transakcí.

Chcete-li zjistit více o rozhraní MQI, použijte následující odkazy:

- "Volání MQI" na stránce 188

- “Volání synchronizačního bodu” na stránce 189
- “Převod dat, datové typy, definice dat a struktury” na stránce 189
- “Soubory typu stub a soubory knihovny produktu IBM WebSphere MQ” na stránce 190
- “Parametry společné pro všechna volání” na stránce 194
- “Určení vyrovnávacích pamětí” na stránce 195
- “Zpracování signálů UNIX and Linux” na stránce 196

Související pojmy

“Připojování k správci front a odpojování od něj” na stránce 198
Chcete-li používat programovací služby produktu WebSphere MQ , musí mít program připojení ke správci front. V této části se dozvíte, jak se připojit k správci front a odpojit je od správce front.

“Otevírání a zavírání objektů” na stránce 206
Tyto informace poskytují náhled na otevření a zavření objektů produktu WebSphere MQ .

“Vložení zpráv do fronty” na stránce 216
V této části se dozvíte, jak vložit zprávy do fronty.

“Získávání zpráv z fronty” na stránce 231
Tyto informace použijte k získání informací o získávání zpráv z fronty.

“Inquaning about a nastavení atributů objektu” na stránce 307
Atributy jsou vlastnosti, které definují charakteristiky objektu produktu WebSphere MQ .

“Potvrzení a zálohování jednotek práce” na stránce 310
Tyto informace popisují, jak potvrdit a zazálohovat všechny zotavitelné operace get a put, které se vyskytly v jednotce práce.

“Spuštění aplikací produktu IBM WebSphere MQ pomocí spouštěčů” na stránce 316
Informace o spouštěčích a o tom, jak spustit aplikace IBM WebSphere MQ pomocí spouštěčů.

“Práce s MQI a klastry” na stránce 332
Existují speciální volby na voláních a návratových kódech, které se vztahují ke klastrování.

Volání MQI

Tyto informace použijte k seznámení s voláními v MQI.

Volání modulu MQI lze seskupit podle následujících kroků:

MQCONN, MQCONNX a MQDISC
Pomocí těchto volání můžete připojit program k (s volbami nebo bez voleb) a odpojit program od správce front. Pokud napíšete programy CICS pro systém z/OS, nemusíte tato volání používat. Doporučuje se je však použít, chcete-li aplikaci portovat na jiné platformy.

MQOPEN a MQCLOSE
Pomocí těchto volání můžete otevřít a zavřít objekt, jako je například fronta.

MQPUT a MQPUT1
Použijte tato volání k vložení zprávy do fronty.

MQGET
Toto volání můžete použít k procházení zpráv ve frontě nebo k odebírání zpráv z fronty.

MQSUB, MQSUBRQ
Pomocí těchto volání zaregistrujte odběr do tématu a vyžádejte publikování odpovídající odběru.

MQINQ
Použijte toto volání k dotazům na atributy objektu.

MQSET
Použijte toto volání k nastavení některých atributů fronty. Atributy jiných typů objektů nelze nastavit.

MQBEGIN, MQCMIT a MQBACK
Použijte tato volání, je-li produkt WebSphere MQ koordinátorem jednotky práce. Funkce MQBEGIN spustí jednotku práce. MQCMIT a MQBACK ukončí pracovní jednotku, a to buď potvrdit, nebo

odvolávání aktualizací provedených během transakce. Jsou použity příkazy vázaného zpracování vázaného zpracování, vázaného zpracování a odvolání.

MQCRTMH, MQBUFMH, MQMBUBUF, MQDLTMH
S použitím těchto volání můžete vytvořit popisovač zprávy, převést popisovač zprávy na vyrovnávací paměť nebo vyrovnávací paměť na popisovač zprávy a odstranit popisovač zprávy.

MQSETTMP, MQINQMP, MQDLTMP
Pomocí těchto volání můžete nastavit vlastnost zprávy na popisovači zprávy, zjišťovat vlastnost zprávy a odstranit vlastnost z manipulátoru zprávy.

MQCB, MQCB_FUNCTION, MQCTL
Tato volání slouží k registraci a řízení funkce zpětného volání.

MQSTAT
Použijte toto volání k načtení informací o stavu pro předchozí asynchronní operace put.

Popis volání MQI naleznete v tématu Popisy volání .

Volání synchronizačního bodu

Tyto informace použijte k vyhledání informací o volání synchronizačního bodu na různých platformách.

Volání bodu synchronizace jsou k dispozici následujícím způsobem:

Volání IBM WebSphere MQ na platformách Windows, UNIX a Linux

Následující produkty poskytují volání MQCMIT a MQBACK:

- IBM WebSphere MQ pro Windows
- IBM WebSphere MQ v systémech UNIX and Linux

Pomocí volání synchronizačních bodů v programech můžete sdělit správci front, že všechny operace MQGET a MQPUT od posledního bodu synchronizace mají být trvale provedeny (potvrzeny) nebo mají být vráceny zpět. Chcete-li potvrdit a zazálohovat změny v prostředí CICS , použijte příkazy jako EXEC CICS SYNCPOINT a EXEC CICS SYNCPOINT ROLLBACK.

Převod dat, datové typy, definice dat a struktury

Tyto informace použijte, chcete-li se dozvědět více o konverzích dat, základních datových typech, definicích dat produktu WebSphere MQ a strukturách při použití rozhraní fronty zpráv.

Převod dat

Volání MQXCNVC (převod znaků) převádí znaková data zprávy z jednoho znaku na jiný. S výjimkou produktu WebSphere MQ for z/OSse toto volání používá pouze z uživatelské procedury pro převod dat.

Viz MQXCNVC-Konvertování znaků pro syntaxi použitou při volání MQXCNVC a “Zápis uživatelských procedur pro převod dat” na stránce 399 pro navádění na psaní a vyvolání uživatelských procedur pro převod dat.

Elementární datové typy

Pro podporované programovací jazyky poskytuje rozhraní MQI elementární datové typy nebo nestrukturovaná pole.

Tyto datové typy jsou plně popsány v části Elementární datové typy.

Definice dat produktu WebSphere MQ

Soubory definic dat dodávané s produktem WebSphere MQ obsahují:

- Definice všech konstant a návratových kódů produktu WebSphere MQ
- Definice struktur a datových typů produktu WebSphere MQ

- Konstantní definice pro inicializaci struktur
- Prototypy funkce pro každý z volání (pouze pro jazyk PL/I a jazyk C)

Úplný popis definičních souborů dat produktu WebSphere MQ naleznete v tématu "Soubory definic dat produktu IBM WebSphere MQ" na stránce 77.

Struktury

Struktury, které se používají s voláními MQI uvedenými v produktu "Volání MQI" na stránce 188, jsou dodávány v souborech definice dat pro každý podporovaný programovací jazyk.

Souhrn struktur najdete v tématu Souhrn datových typů struktury .

Soubory typu stub a soubory knihovny produktu IBM WebSphere MQ

Zde jsou uvedeny programy stubu a soubory knihoven, které jsou zde uvedeny, pro každou platformu.

Další informace o tom, jak používat programy typu stub a soubory knihoven při vytváření spustitelné aplikace, najdete v tématu "Sestavení aplikace IBM WebSphere MQ" na stránce 412. Informace o odkazech na soubory knihovny C + +, viz Použití C++ *WebSphere MQ Using C++*.

IBM WebSphere MQ pro Windows

V systému IBM WebSphere MQ for Windowsje třeba připojit program k souborům knihovny MQI dodaným pro prostředí, ve kterém spouštíte svoji aplikaci, navíc k souborům poskytujícím operační systém:

Tabulka 25. Soubory knihovny pro aplikace systému Windows

Soubor knihovny	**Prostředí**
MQ_INSTALLATION_PATH\Tools\Lib\mqm.lib	Server for C (32bitový)
MQ_INSTALLATION_PATH\Tools\Lib\mqic.lib	Klient pro C (32bitový)
MQ_INSTALLATION_PATH\Tools\Lib\mqmxa.lib	Rozhraní XA serveru pro C (32bitový)
MQ_INSTALLATION_PATH\Tools\Lib\mqcxa.lib	Rozhraní XA klienta pro C (32bitové)
MQ_INSTALLATION_PATH\Tools\Lib\mqicxa.lib	Klient MTS pro C (32bitový)
MQ_INSTALLATION_PATH\Tools\Lib\mqmcics4.lib	Podpora serveru TXSeries CICS pro C (32bitová verze)
MQ_INSTALLATION_PATH\Tools\Lib\mqccics4.lib	Podpora klienta TXSeries CICS pro C (32bitová verze)
MQ_INSTALLATION_PATH\Tools\Lib\mqmzf.lib	Instalovatelné služby jsou ukončeny pro C (32bitový)
MQ_INSTALLATION_PATH\Tools\Lib\mqmcbb.lib	Server pro produkt IBM COBOL (32bitový)
MQ_INSTALLATION_PATH\Tools\Lib\mqmcb.lib	Server pro Micro Focus COBOL (32bitový)
MQ_INSTALLATION_PATH\Tools\Lib\mqiccbb.lib	Klient pro IBM COBOL (32bitový)
MQ_INSTALLATION_PATH\Tools\Lib\mqiccb.lib	Klient pro Micro Focus COBOL (32bitový)
MQ_INSTALLATION_PATH\Tools\Lib\imqs23vn.lib	Server pro C++ (32bitový)
MQ_INSTALLATION_PATH\Tools\Lib\imqc23vn.lib	Klient pro C++ (32bitový)
MQ_INSTALLATION_PATH\Tools\Lib\imqb23vn.lib	Základ pro C++ (32bitový)
MQ_INSTALLATION_PATH\Tools\Lib\imqx23vn.lib	Klient MTS pro C++ (32bitový)
MQ_INSTALLATION_PATH\Tools\Lib64\mqm.lib	Server for C (64bitový)
MQ_INSTALLATION_PATH\Tools\Lib64\mqic.lib	Klient pro C (64bitový)

Tabulka 25. Soubory knihovny pro aplikace systému Windows (pokračování)	
Soubor knihovny	**Prostředí**
MQ_INSTALLATION_PATH\Tools\Lib64\mqmxa.lib	Rozhraní XA serveru pro C (64bitový)
MQ_INSTALLATION_PATH\Tools\Lib64\mqcxa.lib	Rozhraní XA klienta pro C (64 bitů)
MQ_INSTALLATION_PATH\Tools\Lib64\mqicxa.lib	Klient MTS pro C (64bitový)
MQ_INSTALLATION_PATH\Tools\Lib64\mqmcbb.lib	Server pro IBM COBOL (64bitový)
MQ_INSTALLATION_PATH\Tools\Lib64\mqmcb.lib	Server pro Micro Focus COBOL (64bitový)
MQ_INSTALLATION_PATH\Tools\Lib64\mqiccbb.lib	Klient pro IBM COBOL (64bitový)
MQ_INSTALLATION_PATH\Tools\Lib64\mqiccb.lib	Klient pro Micro Focus COBOL (64bitový)
MQ_INSTALLATION_PATH\Tools\Lib64\imqs23vn.lib	Server pro C++ (64bitový)
MQ_INSTALLATION_PATH\Tools\Lib64\imqc23vn.lib	Klient pro C++ (64bitový)
MQ_INSTALLATION_PATH\Tools\Lib64\imqb23vn.lib	Základ pro C++ (64bitový)
MQ_INSTALLATION_PATH\Tools\Lib64\imqx23vn.lib	Klient MTS pro C++ (64bitový)

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

Použijte `amqmdnet.dll` pro kompilaci .NET programů. Další informace naleznete v části “Kompilace programů WebSphere MQ .NET” na stránce 574 v části “Použití .NET” na stránce 540 .

Tyto soubory se dodávají kvůli kompatibilitě s předchozími vydáními:

mqic32.lib
mqic32xa.lib

IBM WebSphere MQ pro systém AIX

V systému IBM WebSphere MQ for AIXmusíte svůj program propojit se soubory knihovny MQI dodanými pro prostředí, ve kterém spouštíte aplikaci, kromě těch, které poskytuje operační systém.

V aplikaci bez podprocesů:

Tabulka 26. Soubory knihovny pro aplikace AIX bez podprocesů	
Soubor knihovny	**Prostředí**
libmqm.a	Server pro C
libmqic.a & **libmqm.a**	Klient pro C
libmqmzf.a	Výstupy instalovatelné služby pro C
libmqmxa.a	Rozhraní XA serveru
libmqmxa64.a	Alternativní rozhraní XA serveru
libmqcxa.a	Rozhraní XA klienta
libmqcxa64.a	Alternativní rozhraní XA klienta

Tabulka 26. Soubory knihovny pro aplikace AIX bez podprocesů (pokračování)	
Soubor knihovny	**Prostředí**
libmqmcbrt.o	Běhová knihovna produktu WebSphere MQ pro podporu Micro Focus COBOL
libmqmcb.a	Server pro COBOL
libmqicb.a	Klient pro COBOL
libimqc23ia.a	Klient pro C++
libimqs23ia.a	Server pro C++

V aplikaci se závitem:

Tabulka 27. Soubory knihovny pro aplikace v systému AIX s podporou podprocesů	
Soubor knihovny	**Prostředí**
libmqm_r.a	Server pro C
libmqic_r.a & **libmqm_r.a**	Klient pro C
libmqmzf_r.a	Výstupy instalovatelné služby pro C
libmqmxa_r.a	Rozhraní XA serveru
libmqmxa64_r.a	Alternativní rozhraní XA serveru
libmqcxa_r.a	Rozhraní XA klienta
libmqcxa64_r.a	Alternativní rozhraní XA klienta
libimqc23ia_r.a	Klient pro C++
libimqs23ia_r.a	Server pro C++

IBM WebSphere MQ pro HP-UX

V systému IBM WebSphere MQ for HP-UXmusíte svůj program propojit se soubory knihovny MQI dodanými pro prostředí, ve kterém spouštíte aplikaci, kromě těch, které poskytuje operační systém.

Platforma IA64 (IPF)

V aplikaci bez podprocesů:

Tabulka 28. Soubory knihovny pro nevláknové aplikace HP-UX	
Soubor knihovny	**Prostředí**
libmqm.so	Server pro C
libmqic.so & **libmqm.so**	Klient pro C
libmqmzf.so	Výstupy instalovatelné služby pro C
libmqmxa.so	Rozhraní XA serveru
libmqmxa64.so	Alternativní rozhraní XA serveru
libmqcxa.so	Rozhraní XA klienta
libmqcxa64.so	Alternativní rozhraní XA klienta
libimqi23ah.so	JAZYK C++

Tabulka 28. Soubory knihovny pro nevláknové aplikace HP-UX (pokračování)	
Soubor knihovny	**Prostředí**
libmqmcbrt.o	Běhová knihovna produktu WebSphere MQ pro podporu Micro Focus COBOL
libmqmcb.so	Server pro COBOL
libmqicb.so	Klient pro COBOL

V aplikaci se závitem:

Tabulka 29. Soubory knihovny pro podprocesové aplikace HP-UX	
Soubor knihovny	**Prostředí**
libmqm_r.so	Server pro C
libmqmzf_r.so & **libmqm_r.so**	Výstupy instalovatelné služby pro C
libmqmxa_r.so	Rozhraní XA serveru
libmqmxa64_r.so	Alternativní rozhraní XA serveru
libmqcxa_r.so	Rozhraní XA klienta
libmqcxa64_r.so	Alternativní rozhraní XA klienta
libimqi23ah_r.so	JAZYK C++

položky IBM WebSphere MQ pro Linux

V produktu IBM WebSphere MQ for Linuxmusíte propojit svůj program se soubory knihovny MQI dodanými pro prostředí, ve kterém spouštíte aplikaci, kromě těch, které poskytuje operační systém.

V aplikaci bez podprocesů:

Tabulka 30. Soubory knihovny pro aplikace bez podprocesů Linux	
Soubor knihovny	**Prostředí**
libmqm.so	Server pro C
libmqic.so & **libmqm.so**	Klient pro C
libmqmzf.so	Výstupy instalovatelné služby pro C
libmqmxa.so	Rozhraní XA serveru
libmqmxa64.so	Alternativní rozhraní XA serveru
libmqcxa.so	Rozhraní XA klienta
libmqcxa64.so	Alternativní rozhraní XA klienta
libimqc23gl.so	Klient pro C++
libimqs23gl.so	Server pro C++

V aplikaci se závitem:

Tabulka 31. Soubory knihovny pro aplikace Linux s podporou podprocesů	
Soubor knihovny	**Prostředí**
libmqm_r.so	Server pro C
libmqic_r.so & **libmqm_r.so**	Klient pro C

Tabulka 31. Soubory knihovny pro aplikace Linux s podporou podprocesů (pokračování)	
Soubor knihovny	**Prostředí**
libmqmzf_r.so	Výstupy instalovatelné služby pro C
libmqmxa_r.so	Rozhraní XA serveru
libmqmxa64_r.so	Alternativní rozhraní XA serveru
libmqcxa_r.so	Rozhraní XA klienta
libmqcxa64_r.so	Alternativní rozhraní XA klienta
libimqc23gl_r.so	Klient pro C++
libimqs23gl_r.so	Server pro C++

IBM WebSphere MQ pro Solaris

V produktu IBM WebSphere MQ for Solaris musíte propojit svůj program se soubory knihovny MQI dodanými pro prostředí, ve kterém spouštíte svou aplikaci, kromě těch, které jsou poskytovány operačním systémem.

Tabulka 32. Soubory knihovny pro aplikace systému Solaris	
Soubor knihovny	**Prostředí**
libmqm.so	Server a klient pro C
libmqmzse.so	Pro C
libmqic.so	Klient pro C
libmqmcs.so	Běžné služby pro jazyk C
libmqmzf.so	Výstupy instalovatelné služby pro C
libmqmxa.so	Rozhraní XA serveru
libmqmxa64.so	Alternativní rozhraní XA serveru
libmqcxa.so	Rozhraní XA klienta
libmqcxa64.so	Alternativní rozhraní XA klienta
libimqc23as.a	Klient pro C++
libimqs23as.a	Server pro C++

Parametry společné pro všechna volání

Pro všechna volání jsou společné dva typy parametrů: popisovače a návratové kódy.

Použití popisovačů

Všechna volání MQI používají jeden nebo více *manipulátorů*. Ty identifikují správce front, frontu nebo jiný objekt, zprávu nebo odběr, jak je to vhodné pro volání.

Aby program mohl komunikovat se správcem front, musí mít tento program jedinečný identifikátor, v němž zná správce front. Tento identifikátor se nazývá *manipulátor připojení*, někdy označovaný také jako *Hconn*. Pro programy CICS je manipulátor připojení vždy nula. Pro všechny ostatní platformy nebo styly programů je manipulátor připojení vrácen voláním MQCONN nebo MQCONNX při připojování programu ke správci front. Programy předávají obslužnou rutinu připojení jako vstupní parametr, používají-li jiná volání.

Aby mohl program pracovat s objektem produktu WebSphere MQ , musí mít tento program jedinečný identifikátor, který tento objekt zná. Tento identifikátor se nazývá *popisovač objektu*, někdy označovaný také jako *Hobj*. Manipulátor je vrácen voláním MQOPEN, když program otevře objekt pro práci s ním.

Programy předávají obslužnou rutinu objektu jako vstupní parametr, když používají následné volání MQPUT, MQGET, MQINQ, MQSET nebo MQCLOSE.

Podobně volání MQSUB vrací *popisovač odběru* nebo *Hsub*, který se používá k identifikaci odběru v následných voláních MQGET, MQCB nebo MQSUBRQ a určité výzvy ke zpracování vlastností zpráv používají *popisovač zprávy* nebo *Hmsg*.

Vysvětlení návratových kódů

Kód dokončení a kód příčiny jsou vráceni jako výstupní parametry každým voláním. Ty jsou souhrnně označovány jako *návratové kódy*.

Chcete-li zobrazit, zda je volání úspěšné, každé volání vrátí při dokončení volání *kód dokončení* . Kód dokončení je obvykle buď MQCC_OK označující úspěch, nebo MQCC_FAILED indikující selhání. Některá volání mohou vrátit přechodný stav, MQCC_WARNING, označující dílčí úspěch.

Každé volání také vrací *kód příčiny* , který zobrazuje příčinu selhání nebo částečného úspěchu volání. Existuje mnoho kódů příčiny, které se za takových okolností vztahují k zaplnění fronty, operace získání není pro frontu povolena a pro správce front není definována konkrétní fronta. Programy mohou použít kód příčiny k rozhodnutí, jak pokračovat. Mohou například vyzvat uživatele ke změně svých vstupních dat, pak znovu zavolat, nebo mohou vrátit chybovou zprávu uživateli.

Je-li kód dokončení MQCC_OK, kód příčiny je vždy MQRC_NONE.

Vyplnění a kódy příčiny pro každé volání jsou uvedeny s popisem tohoto volání. Viz Call descriptions a vyberte příslušné volání ze seznamu.

Podrobnější informace, včetně nápadů pro nápravnou akci, najdete v tématu:

- Kódy příčin pro všechny ostatní platformy WebSphere MQ

Určení vyrovnávacích pamětí

Správce front se odkazuje na vyrovnávací paměti pouze v případě, že jsou vyžadovány. Pokud nevyžadujete vyrovnávací paměť při volání nebo vyrovnávací paměť má nulovou délku, můžete použít ukazatel null na vyrovnávací paměť.

Při zadávání velikosti vyrovnávací paměti, kterou vyžadujete, vždy použijte délku dat.

Použijete-li vyrovnávací paměť k uložení výstupu z volání (například k zadržení dat zprávy pro volání MQGET nebo hodnoty atributů dotazovaných voláním MQINQ), správce front se pokusí o vrácení kódu příčiny, pokud zadaná vyrovnávací paměť není platná nebo je v úložišti jen pro čtení. Avšak nemusí být vždy schopen vrátit kód příčiny.

Aspekty produktu UNIX and Linux

Pokyny, které byste měli znát.

Při vývoji aplikací produktu UNIX and Linux vezměte na vědomí následující body.

Systémové volání fork v systémech UNIX and Linux

Všimněte si těchto pokynů při použití systémového volání fork v aplikacích produktu IBM WebSphere MQ .

Chce-li vaše aplikace používat produkt `fork`, nadřízený proces této aplikace by měl volat `fork` před provedením jakýchkoli volání IBM WebSphere MQ , například `MQCONN`, nebo vytvořením objektu IBM WebSphere MQ pomocí **`ImqQueueManager`**.

Pokud chce vaše aplikace po provedení libovolných volání IBM WebSphere MQ vytvořit podřízený proces, musí kód aplikace používat `fork()` s produktem `exec()` , aby bylo zajištěno, že je podřízeným prvkem nová instance a ne přesná kopie nadřízené položky.

Pokud vaše aplikace nevyužívá produkt `exec()`, volání rozhraní API produktu IBM WebSphere MQ provedené v podřízeném procesu vrátí funkci MQRC_ENVIRONMENT_ERROR.

Zpracování signálů UNIX and Linux

Toto neplatí pro produkt WebSphere MQ for z/OS nebo WebSphere MQ for Windows.

Obecně platí, že systémy UNIX, Linux a IBM i se přesunuly z prostředí bez podprocesů (procesů) do vícevláknového prostředí. V prostředí bez podprocesů mohou být některé funkce implementovány pouze pomocí signálů, ačkoli většina aplikací nemusí být informována o signálech a zpracování signálů. Ve vícevláknovém prostředí podporují vlákna založená na vláknech některé z funkcí, které se používají k implementaci v prostředí bez vláken pomocí signálů.

V mnoha případech se signály a zpracování signálů, i když jsou podporovány, nehodí do vícevláknového prostředí a existují různá omezení. To může být problematické, když integrujete kód aplikace s různými knihovnami middlewaru (spuštěnými jako součást aplikace) v prostředí s podporou podprocesů, kde se každý snaží ošetřit signály. Tradiční přístup k ukládání a obnově obslužných rutin signálů (definovaných pro každý proces), který fungoval, když v rámci procesu existoval pouze jeden podproces, nefunguje v prostředí s více podprocesy. Důvodem je to, že mnoho podprocesů provedení se může pokusit o uložení a obnovení prostředku v rámci celého procesu s nepředvídatelnými výsledky.

Aplikace bez podprocesů
Nevztahuje se na systém Solaris, protože všechny aplikace jsou považovány za závity, i když používají pouze jeden podproces.

Každá funkce MQI nastavuje svůj vlastní popisovač signálu pro signály:

SIGNALRM
SIGBUS
SIGFPE
SIGSEGV
SIGILL

Popisovače uživatelů pro tyto úlohy budou nahrazeny po dobu trvání volání funkce MQI. Další signály mohou být zachyceny běžným způsobem uživatelem napsanými obslužnými rutinami. Pokud obslužnou rutinu nenainstalujete, budou na místě ponechány výchozí akce (například ignorování, výpis jádra nebo ukončení).

Poté, co produkt WebSphere MQ zpracuje synchronní signál (SIGSEGV, SIGBUS, SIGFPE, SIGILL), pokusí se před voláním funkce MQI předat signál k libovolnému registrovanému popisovači signálu.

Aplikace s podprocesy
Podproces je považován za připojený k produktu WebSphere MQ z MQCONN (nebo MQCONNX) do operace MQDISC.

Synchronní signály

Synchronní signály vznikají ve specifickém podprocesu.

Systémy UNIX and Linux bezpečně umožňují nastavení ovladače signálu pro takové signály pro celý proces. Produkt WebSphere MQ však nastavuje vlastní obslužnou rutinu pro následující signály, v rámci procesu aplikace, zatímco jakýkoli podproces je připojen k produktu WebSphere MQ:

SIGBUS
SIGFPE
SIGSEGV
SIGILL

Pokud zapisujete vícevláknové aplikace, existuje pro každý signál pouze jedna obslužná rutina signálu celého procesu. Když produkt WebSphere MQ nastaví vlastní synchronní obslužné rutiny signálu, uloží všechny dříve registrované obslužné rutiny pro každý signál. Poté, co produkt WebSphere MQ zpracovává některý z uvedených signálů, produkt WebSphere MQ se pokusí o volání obslužné rutiny signálu, která byla v platnosti v době prvního připojení WebSphere MQ v rámci procesu. Dříve registrované obslužné rutiny jsou obnoveny, když se všechny podprocesy aplikace odpojují od produktu WebSphere MQ.

Vzhledem k tomu, že obslužné rutiny signálů jsou ukládány a obnovovány produktem WebSphere MQ, nesmí podprocesy aplikací pro tyto signály vytvářet obslužné rutiny signálů, zatímco existuje možnost, že je k produktu WebSphere MQpřipojen další podproces stejného procesu.

Poznámka: Když aplikace nebo knihovna middlewaru (spuštěná jako část aplikace) zřídí obslužnou rutinu signálu, když je podproces připojen k produktu WebSphere MQ, musí obslužná rutina signálu aplikace během zpracování tohoto signálu zavolat odpovídající obslužnou rutinu WebSphere MQ .

Při vytváření a obnově obslužných rutin signálu musí být hlavní zásadou, že poslední manipulační program signálu, který má být uložen, musí být první, který se má obnovit:

- Když aplikace zřídí popisovač signálu po připojení k produktu WebSphere MQ, musí být předchozí popisovač signálu obnoven dříve, než se aplikace odpojí od WebSphere MQ.
- Když aplikace zavede manipulační program signálu před připojením k produktu WebSphere MQ, musí se aplikace od produktu WebSphere MQ odpojit před obnovením jeho obslužné rutiny signálu.

Poznámka: Selhání při sledování obecné zásady, že poslední obslužná rutina signálu, která má být uložena, musí být první, která má být obnovena, může vést k neočekávanému zpracování signálů v aplikaci a potenciálně i ztrátě signálů aplikací.

Asynchronní signály

Produkt WebSphere MQ nepoužívá žádné asynchronní signály v aplikacích s podporou podprocesů, pokud se nejedná o klientské aplikace.

Další pokyny pro klientské aplikace s podporou podprocesů

Produkt WebSphere MQ zpracovává následující signály během I/O na server. Tyto signály jsou definovány v komunikačním zásobníku. Aplikace nesmí vytvořit obslužnou rutinu signálů pro tyto signály, zatímco je podproces připojen ke správci front:

SIGPIPE (pro TCP/IP)

Další pokyny
Všimněte si těchto pokynů při použití zpracování signálů systému UNIX .

Aplikace Fastpath (důvěryhodné)

Aplikace Fastpath běží ve stejném procesu jako WebSphere MQ a tak jsou spuštěny v prostředí s podporou podprocesů.

V tomto prostředí produkt WebSphere MQ zpracovává synchronní signály SIGSEGV, SIGBUS, SIGFPE a SIGILL. Všechny ostatní signály nesmí být během připojení k produktu WebSphere MQdoručeny do aplikace Fastpath. Místo toho musí být blokovány nebo zpracovány aplikací. Pokud aplikace Fastpath zadrží takovou událost, musí být správce front zastaven a restartován, nebo může být ponechán v nedefinovaném stavu. Úplný seznam omezení pro aplikace Fastpath pod MQCONNX viz “Připojení ke správci front pomocí volání MQCONNX” na stránce 201.

Volání funkcí MQI v obslužných rutinách signálů

Když jste v popisovači signálu, nevolejte funkci MQI.

Pokusíte-li se volat funkci MQI z obslužné rutiny signálu při zpracování jiné funkce MQI, vrátí se hodnota MQRC_CALL_IN_PROGRESS. Pokud se pokusíte volat funkci MQI z obslužné rutiny signálu a přitom není aktivní žádná jiná funkce MQI, bude pravděpodobně někdy během operace selhat kvůli omezením operačního systému, kdy lze z obslužné rutiny vydat pouze výběrová volání.

Pro metody destruktoru C + +, které mohou být volány automaticky během ukončení programu, možná nebudete moci zastavit volání funkcí MQI. Ignorujte všechny chyby týkající se MQRC_CALL_IN_PROGRESS. Pokud obslužná rutina signálu zavolá funkci exit (), WebSphere MQ zazálohuje nepotvrzené zprávy v synchronizačním bodě jako obvykle a zavře všechny otevřené fronty.

Signály během volání MQI

Funkce MQI nevracejí kód EINTR ani žádný ekvivalent aplikačních programů.

Pokud během volání MQI dojde k signálu a volání obslužné rutiny volá *return*, volání pokračuje v tom, jako by se signál nestal. Příkaz MQGET nemůže být zejména přerušen signálem k okamžitému vrácení řízení do aplikace. Chcete-li přerušit MQGET, nastavte frontu na volbu GET_DISABLED; alternativně použijte smyčku kolem volání MQGET s konečným časovým intervalem (MQGMO_WAIT s parametrem gmo.WaitInterval) a pomocí obslužné rutiny signálu (v prostředí bez podprocesů) nebo ekvivalentní funkce v prostředí s podprocesy nastavte příznak, který přeruší smyčku.

V prostředí AIX produkt WebSphere MQ vyžaduje restartování systémových volání přerušených signály. Při vytváření vašeho vlastního popisovače signálu pomocí funkce sigaction (2) nastavte příznak SA_RESTART v poli sa_flags nové struktury akce jinak WebSphere MQ nemůže být schopen dokončit volání přerušené signálem.

Uživatelské procedury a instalovatelné služby

Uživatelské procedury a instalovatelné služby, které se spouštějí jako součást procesu WebSphere MQ v prostředí s podporou podprocesů, mají stejná omezení jako v případě aplikací se zkrácenou cestou. Považujte je za trvale připojené k produktu WebSphere MQ a nepoužívají tak signály nebo podprocesy operačního systému, které nejsou zabezpečeny vlákny.

Obslužné rutiny výstupních bodů

Uživatelé mohou instalovat obslužné rutiny ukončení pro aplikaci WebSphere MQ pomocí systémové služby produktu **SYS$DCLEXH** .

Obslužná rutina ukončení obdrží řízení, když se obrázek ukončí. K ukončení obrazu se obvykle dojde, když voláte službu Konec ($EXIT) nebo Vynutit ukončení ($FORCEX). $FORCEX přeruší cílový proces v uživatelském režimu. Poté všechny obslužné rutiny uživatelských procedur v uživatelském režimu (zavedené $DCLEXH) se začnou provádět v opačném pořadí, než je zařízení. Další informace o ovladačích ukončení a $FORCEX najdete v příručce *VMS Programming Concepts Manual* a *VMS System Services Manual*.

Vočíte-li funkci MQI z obslužné rutiny ukončení, chování funkce závisí na způsobu, jakým byl obraz ukončen. Pokud byl obraz ukončen, zatímco je aktivní jiná funkce MQI, je vrácena hodnota MQRC_CALL_IN_PROGRESS .

Je možné volat funkci MQI z obslužné rutiny ukončení, pokud není aktivní žádná jiná funkce MQI a jsou zakázána volání upcall pro aplikaci produktu WebSphere MQ . Pokud jsou povolena pro aplikaci WebSphere MQ volání upcalls, dojde k selhání s kódem příčiny MQRC_HCONN_ERROR.

K rozsahu volání MQCONN nebo MQCONNX se obvykle používá podproces, který jej vydal. Pokud jsou povolena volání upcalls, obslužná rutina ukončení se spustí jako oddělený podproces a popisovače připojení nemohou být sdíleny.

Obslužné rutiny ukončení jsou spuštěny v rámci přerušeného kontextu cílového procesu. Je na aplikaci, aby bylo zajištěno, že akce provedené obslužnou rutinou jsou bezpečné a spolehlivé, pro asynchronně přerušovaný kontext, ze kterého jsou volány.

Připojování k správci front a odpojování od něj

Chcete-li používat programovací služby produktu WebSphere MQ , musí mít program připojení ke správci front. V této části se dozvíte, jak se připojit k správci front a odpojit je od správce front.

Způsob, jakým toto připojení závisí, závisí na platformě a prostředí, ve kterém tento program funguje:

z/OS batch, WebSphere MQ for IBM i, WebSphere MQ on UNIX systems, WebSphere MQ on Linux systems, and WebSphere MQ for Windows

Programy, které jsou spuštěny v těchto prostředích, mohou používat volání MQCONN MQI k připojení a volání MQDISC, které se má odpojit od správce front. Alternativně mohou programy používat volání MQCONNX.

Dávkové programy z/OS mohou ve stejné TCB navázat spojení, postupně nebo souběžně s více správci front.

IMS
Řídicí region IMS je připojen k jednomu nebo více správcům front při spuštění. Toto připojení je řízeno příkazy IMS . Nicméně autoři zpráv front zpráv IMS musí pomocí volání MQI MQCONN určit správce front, ke kterému se mají připojit. K odpojení od tohoto správce front mohou použít volání MQDISC.

Po volání funkce IMS , která zřídí synchronizační bod, a před zpracováním zprávy pro jiného uživatele adaptér IMS zajistí, aby aplikace zavřela a odpojuje od správce front.

Programy IMS se mohou souběžně nebo souběžně připojit k více správcům front ve stejné TCB.

CICS Transaction Server pro z/OS a CICS pro MVS/ESA
Programy CICS nemusí provádět žádnou práci pro připojení ke správci front, protože je připojen samotný systém CICS . Toto připojení se obvykle provádí automaticky při inicializaci, ale můžete také použít transakci CKQC, která jedodáván s produktem WebSphere MQ for z/OS.

Úlohy CICS se mohou připojit pouze ke správci front, k němuž je připojen samotný region CICS .

Poznámka: Programy CICS mohou také používat volání connect a disconnect MQI (MQCONN a MQDISC). Možná budete chtít toto provést, abyste mohli tyto aplikace portovat do jiných prostředí než CICS s minimem změn. Nicméně tato volání *vždy* budou 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 podporováno pouze jedno připojení v daném okamžiku. Aplikace CICS musí při volání manipulátoru připojení a volání MQDISC před ukončením volání funkce MQCONN vydat volání MQCONN.

Chcete-li zjistit více o připojení a odpojení od správce front, použijte následující odkazy:

- “Připojení ke správci front pomocí volání MQCONN” na stránce 199
- “Připojení ke správci front pomocí volání MQCONNX” na stránce 201
- “Odpojení programů od správce front pomocí příkazu MQDISC” na stránce 205

Související pojmy
“Přehled rozhraní fronty zpráv” na stránce 187
Zde jsou uvedeny informace o komponentách rozhraní MQI (Message Queue Interface).

“Otevírání a zavírání objektů” na stránce 206
Tyto informace poskytují náhled na otevření a zavření objektů produktu WebSphere MQ .

“Vložení zpráv do fronty” na stránce 216
V této části se dozvíte, jak vložit zprávy do fronty.

“Získávání zpráv z fronty” na stránce 231
Tyto informace použijte k získání informací o získávání zpráv z fronty.

“Inquaning about a nastavení atributů objektu” na stránce 307
Atributy jsou vlastnosti, které definují charakteristiky objektu produktu WebSphere MQ .

“Potvrzení a zálohování jednotek práce” na stránce 310
Tyto informace popisují, jak potvrdit a zazálohovat všechny zotavitelné operace get a put, které se vyskytly v jednotce práce.

“Spuštění aplikací produktu IBM WebSphere MQ pomocí spouštěčů” na stránce 316
Informace o spouštěčích a o tom, jak spustit aplikace IBM WebSphere MQ pomocí spouštěčů.

“Práce s MQI a klastry” na stránce 332
Existují speciální volby na voláních a návratových kódech, které se vztahují ke klastrování.

Připojení ke správci front pomocí volání MQCONN

Pomocí těchto informací se naučíte, jak se připojit ke správci front pomocí volání MQCONN.

Obecně lze připojit buď ke specifickému správci front, nebo k výchozímu správci front:

- Pro produkt IBM WebSphere MQ for z/OSje v dávkovém prostředí určen výchozí správce front v modulu CSQBDEFV.
- Pro systémy IBM WebSphere MQ pro systémy Windows, IBM i, UNIXa Linux je výchozí správce front určen v souboru mqs.ini .

Alternativně lze v prostředí z/OS MVS dávek TSO a RRS připojit se k libovolnému správci front v rámci skupiny sdílení front. Požadavek MQCONN nebo MQCONNX vybere jednoho z aktivních členů skupiny.

Když se připojíte ke správci front, musí být lokální pro danou úlohu. Musí náležet do stejného systému jako aplikace IBM WebSphere MQ .

V prostředí IMS musí být správce front připojen k řídicí oblasti IMS a k závislé oblasti, kterou tento program používá. Výchozí správce front je určen v modulu CSQQDEFV, je-li nainstalován produkt IBM WebSphere MQ for z/OS .

S prostředím TXSeries CICS a TXSeries pro produkty Windows a AIXmusí být správce front definován jako prostředek XA pro produkt CICS.

Chcete-li se připojit k výchozímu správci front, volejte MQCONN a uveďte název sestávající zcela z mezer nebo začněte znakem null (X'00 ').

Aplikace musí být autorizována, aby se mohla úspěšně připojit ke správci front. Další informace najdete v tématu Zabezpečení.

Výstup z MQCONN je:

- Popisovač připojení (**Hconn**).
- Kód dokončení
- Kód příčiny

Použít obslužnou rutinu připojení při následných voláních MQI.

Pokud kód příčiny informuje o tom, že aplikace je již připojena k tomuto správci front, je vrácený popisovač připojení stejný jako ten, který byl vrácen při prvním připojení aplikace. Aplikace nesmí v této situaci vyvolat volání MQDISC, protože volající aplikace očekává, že zůstane připojena.

Rozsah manipulátoru připojení je stejný jako obor manipulátoru objektu (viz “Otevírání objektů pomocí volání MQOPEN” na stránce 207).

Popisy parametrů jsou uvedeny v popisu volání MQCONN v MQCONN.

Volání MQCONN selže, pokud je správce front při zadávání volání do klidového stavu nebo pokud se správce front vypíná.

Rozsah MQCONN nebo MQCONNX

K rozsahu volání MQCONN nebo MQCONNX se obvykle používá podproces, který jej vydal. To znamená, že manipulátor připojení vrácený z volání je platný pouze v rámci podprocesu, který vydal volání. V jednom okamžiku může být pomocí manipulátoru provedeno pouze jedno volání. Je-li použit z jiného podprocesu, je odmítnut jako neplatný. Máte-li ve své aplikaci více podprocesů a každý chce používat volání IBM WebSphere MQ , musí každý z nich zadat volání MQCONN nebo MQCONNX.

Pokud proces provádí více volání MQCONN, není třeba pro každé volání, které má být provedeno do stejného správce front. Z podprocesu však lze v daném okamžiku provádět pouze jedno připojení produktu WebSphere MQ . Případně zvažte použití produktu “Sdílená připojení (nezávislá na podprocesech) s MQCONNX” na stránce 204 k povolení více připojení produktu WebSphere MQ z jednoho podprocesu a připojení produktu WebSphere MQ , které má být použito z libovolného podprocesu.[1]

[1] Při použití aplikací s podporou podprocesů s produktem IBM WebSphere MQ v systémech UNIX and Linux je třeba zajistit, aby aplikace měly dostatečnou velikost zásobníku pro podprocesy. Zvažte použití velikosti zásobníku 256 KB nebo větší, když aplikace s podporou podprocesů provádějí volání MQI buď samostatně, nebo s jinými obslužnými rutinami signálu (například CICS).

Je-li vaše aplikace spuštěna jako klient, může se připojit k více než jednomu správci front v rámci podprocesu.

Připojení ke správci front pomocí volání MQCONNX

Volání MQCONNX je podobné volání MQCONN, ale obsahuje volby pro řízení toho, jak volání funguje.

Jako vstup pro MQCONNX můžete zadat název správce front nebo název skupiny sdílení front v systémech sdílených front systému z/OS . Výstup z MQCONNX je:

- Popisovač připojení (Hconn)
- Kód dokončení
- Kód příčiny

Manipulátor připojení se používá při následných voláních MQI.

Popis všech parametrů MQCONNX je uveden v MQCONNX. Pole *Options* umožňuje nastavit parametry STANDARD_BINDING, FASTPATH_BINDING, SHARED_BINDING nebo ISOLATED_BINDING pro libovolnou verzi MQCNO. Pomocí volání MQCONNX můžete také vytvořit sdílená připojení (nezávislé na podprocesech). Další informace o těchto tématech viz “Sdílená připojení (nezávislá na podprocesech) s MQCONNX” na stránce 204 .

VAZBA MQCNO_STANDARD_BINDING

MQCONNX (například MQCONN) je standardně tvořen dvěma logickými podprocesy, ve kterých se aplikace WebSphere MQ a lokální agent správce front spouštějí v samostatných procesech. Aplikace WebSphere MQ požádá o provedení operace WebSphere MQ a lokálního agenta správce front. Toto je definováno volbou MQCNO_STANDARD_BINDING na volání MQCONNX.

Pokud zadáte MQCNO_STANDARD_BINDING, volání MQCONNX používá buď MQCNO_SHARED_BINDING nebo MQCNO_ISOLATED_BINDING, v závislosti na hodnotě atributu typu DefaultBindsprávce front, který je definován v souboru qm.ini nebo v registru systému Windows .

Toto je výchozí hodnota.

Pokud odkazujete na knihovnu produktu `mqm` , je nejprve proveden pokus o použití standardního připojení k serveru s použitím výchozího typu vazby. Pokud se základní knihovna serveru nepodařilo načíst, bude namísto toho proveden pokus o připojení klienta.

- Je-li zadána proměnná prostředí MQ_CONNECT_TYPE, může být dodána jedna z následujících voleb pro změnu chování MQCONN nebo MQCONNX, je-li zadáno MQCNO_STANDARD_BINDING. (Výjimkou je, je-li hodnota MQCNO_FASTPATH_BINDING zadána s hodnotou MQ_CONNECT_TYPE nastaveným na hodnotu LOCAL nebo STANDARD , aby umožnila snížení úrovně zkrácené cesty administrátorem bez související změny aplikace:

Hodnota	**Význam**
CLIENT	Pokus o připojení klienta se provede pouze.
Rychlý	Tato hodnota byla v předchozích verzích podporována, ale je nyní ignorována, pokud byla zadána.
LOKÁLNÍ	Pokus o připojení k serveru se pouze pokusil. Připojení zrychleného přístupu se sníží na standardní připojení k serveru.
STANDARD	Podporováno z důvodu kompatibility s předchozími verzemi. Tato hodnota je nyní považována za LOCAL.

- Není-li proměnná prostředí MQ_CONNECT_TYPE nastavena při volání MQCONN, je proveden pokus o připojení standardního serveru s použitím výchozího typu vazby. Pokud se knihovna serveru nepodaří načíst, dojde k pokusu o připojení klienta.

VAZBA MQCNO_FASTPATH_BINDING

Důvěryhodné aplikace znamenají, že se aplikace WebSphere MQ a lokální agent správce front stanou stejným procesem. Vzhledem k tomu, že proces agenta již nemusí pro přístup ke správci front používat rozhraní, tyto aplikace se stanou rozšířením správce front. Toto je definováno volbou MQCNO_FASTPATH_BINDING v rámci volání MQCONNX.

Musíte propojit důvěryhodné aplikace se závitem WebSphere MQ vlákny. Pokyny, jak nastavit aplikaci WebSphere MQ jako důvěryhodná, najdete v tématu Volby MQCNO.

Tato volba poskytuje nejvyšší výkon.

Poznámka: Tato volba ohrožuje integritu správce front: neexistuje žádná ochrana proti přepsání jeho paměti. To platí také v případě, že aplikace obsahuje chyby, které mohou být vystaveny i zprávám a dalším datům ve správci front. Před použitím této volby zvažte tyto problémy.

CQCNO_SHARED_BINDING

Tuto volbu určete, chcete-li aplikaci a lokálního agenta správce front spouštět v samostatných procesech. Tím je zachována integrita správce front, tj. chrání správce front před chybnými programy. Avšak aplikace a agent local-queue-manager sdílejí některé prostředky.

Tato volba je mezilehlá mezi MQCNO_FASTPATH_BINDING a MQCNO_ISOLATED_BINDING, a to jak z hlediska ochrany integrity správce front, tak i z hlediska výkonu volání MQI.

Objekt MQCNO_SHARED_BINDING je ignorován, pokud správce front nepodporuje tento typ vazby. Zpracování pokračuje, jako by tato volba nebyla uvedena.

Pokud se aplikace připojila k lokálnímu správci front pomocí MQCNO_SHARED_BINDING, může být správce front zastaven za běhu aplikace. Pokud správce front restartujete v době, kdy je aplikace stále spuštěna, dojde k selhání pokusu o spuštění správce front s chybou AMQ7018 , protože aplikace stále zadržuje prostředky, které potřebuje správce front.

Chcete-li spustit správce front, je třeba aplikaci ukončit.

VAZBA MQCNO_ISOLATED_BINDING

Zadáním této volby provedete běh aplikace a lokálního agenta správce front v oddělených procesech, jako například pro MQCNO_SHARED_BINDING. V tomto případě je však aplikační proces a agent local-queue-manager izolováni od sebe navzájem, protože nesdílejí prostředky.

Toto je nejbezpečnější volba pro ochranu integrity správce front, ale poskytuje nejpomalejší výkon volání MQI.

Objekt MQCNO_ISOLATED_BINDING je ignorován, pokud správce front nepodporuje tento typ vazby. Zpracování pokračuje, jako by tato volba nebyla uvedena.

VÁZÁNÍ MQCNO_CLIENT_BINDING

Tuto volbu uveďte, chcete-li aplikaci pokusit pouze o připojení klienta. Tato volba má následující omezení:

- Hodnota MQCNO_CLIENT_BINDING byla v systému z/OS odmítnuta s chybou MQRC_OPTIONS_ERROR.
- Hodnota MQCNO_CLIENT_BINDING byla odmítnuta s chybou MQRC_OPTIONS_ERROR, je-li zadána s jinou volbou vazby MQCNO, než je MQCNO_STANDARD_BINDING.
- MQCNO_CLIENT_BINDING není k dispozici pro produkt Java , protože má vlastní mechanizmy pro výběr typu vazby.
- V 7.5.0.7 Před IBM WebSphere MQ Version 7.5.0, opravná sada 7není MQCNO_CLIENT_BINDING dostupné pro prostředí .NET, protože má vlastní mechanizmy pro výběr typu vazby. V produktu Version 7.5.0, Fix Pack 7je omezení pro použití .NET pro MQCNO_CLIENT_BINDING odebráno.

- Není-li proměnná prostředí MQ_CONNECT_TYPE nastavena při volání MQCONNX, je proveden pokus o standardní připojení k serveru s použitím výchozího typu vazby. Pokud se knihovna serveru nepodaří načíst, dojde k pokusu o připojení klienta.

MQCNO_LOCAL_BINDING.

Tuto volbu uveďte, chcete-li provést pokus aplikace o připojení k serveru. Je-li také zadán buď MQCNO_FASTPATH_BINDING, MQCNO_ISOLATED_BINDING nebo MQCNO_SHARED_BINDING, bude místo toho připojení tohoto typu a v této sekci je zdokumentováno. Jinak se provede pokus o standardní připojení k serveru s použitím výchozího typu vazby. Objekt MQCNO_LOCAL_BINDING má následující omezení:

- Hodnota MQCNO_LOCAL_BINDING je v systému z/OSignorována.
- MQCNO_LOCAL_BINDING byl odmítnut s chybou MQRC_OPTIONS_ERROR, pokud je zadán spolu s jinou volbou MQCNO reconnect jiným než MQCNO_RECONNECT_AS_DEF.
- MQCNO_LOCAL_BINDING není k dispozici pro produkt Java , protože má vlastní mechanizmy pro výběr typu vazby.
- V 7.5.0.7 Před produktem IBM WebSphere MQ Version 7.5.0, opravná sada 7není MQCNO_LOCAL_BINDING dostupné pro prostředí .NET, protože má vlastní mechanizmy pro výběr typu vazby. V produktu Version 7.5.0, Fix Pack 7je omezení pro použití .NET pro MQCNO_LOCAL_BINDING odebráno.
- Není-li proměnná prostředí MQ_CONNECT_TYPE nastavena při volání MQCONNX, je proveden pokus o standardní připojení k serveru s použitím výchozího typu vazby. Pokud se knihovna serveru nepodaří načíst, dojde k pokusu o připojení klienta.

V systému z/OS jsou tyto volby tolerovány, ale provádí se pouze standardní vázané připojení. MQCNO verze 3, pro operační systém z/OS, umožňuje čtyři alternativní možnosti:

MQCNO_SERIALIZE_CONN_TAG_QSG

To umožňuje aplikaci požadovat, aby v jedné skupině sdílení front byla současně spuštěna pouze jedna instance aplikace v rámci skupiny sdílení front. Toho lze dosáhnout registrací použití značky připojení s hodnotou, která je určena nebo odvozena z aplikace. Značka je 128bajtový znakový řetězec určený ve verzi 3 MQCNO.

MQCNO_RESTRICT_CONN_TAG_QSG

Tato hodnota se používá v případech, kdy se aplikace skládá z více než jednoho procesu (nebo z TCB), z nichž každý se může připojit ke správci front. Připojení je povoleno pouze v případě, že neexistuje žádné aktuální použití značky, nebo žádost o aplikaci se nachází ve stejném rozsahu zpracování. Jedná se o adresní prostor MVS v rámci stejné skupiny sdílení front jako vlastník značky.

MQCNO_SERIALIZE_CONN_TAG_Q_MGR

Je to podobné jako MQCNO_SERIALIZE_CONN_TAG_QSG, ale pouze lokální správce front je dotazován, aby zjistil, zda již požadovaná značka již je používána.

MQCNO_RESTRICT_CONN_TAG_Q_MGR

Tento postup je podobný objektu MQCNO_RESTRICT_CONN_TAG_QSG, ale pouze lokální správce front je dotazován, aby bylo možné zjistit, zda již požadovaná značka již je používána.

Omezení pro důvěryhodné aplikace

Pro důvěryhodné aplikace platí následující omezení:

- Důvěryhodné aplikace je třeba explicitně odpojit od správce front.
- Před ukončením správce front s příkazem endmqm musíte zastavit důvěryhodné aplikace.
- Musíte použít asynchronní signály a přerušení časovače (například `sigkill`) s MQCNO_FASTPATH_BINDING.
- Na všech platformách se podproces v rámci důvěryhodné aplikace nemůže připojit ke správci front, zatímco jiný podproces ve stejném procesu je připojen k jinému správci front.

- V systému WebSphere MQ v systémech UNIX and Linux je třeba pro všechna volání MQI použít hodnotu mqm jako efektivní userID a groupID . Tato ID můžete změnit před provedením volání mimo rozhraní MQI, které vyžaduje ověření (například otevření souboru), ale *musíte* před provedením dalšího volání MQI změnit zpět na skupinu mqm.
- V produktu WebSphere MQ for HP-UXje pravděpodobné, že aplikace s podporou podprocesů s podporou podprocesů budou pravděpodobně muset nastavit větší velikost zásobníku, než je výchozí. Použijte velikost 256 kB.
- V produktu WebSphere MQ pro důvěryhodné 64bitové aplikace produktu Windows nejsou podporovány 64bitové aplikace. Pokusíte-li se spustit důvěryhodnou 64-bitovou aplikaci, bude snížena úroveň na standardní vázané připojení.
- V produktu WebSphere MQ v systémech UNIX and Linux se nedůvěryhodné 32bitové aplikace nepodporují. Pokusíte-li se spustit důvěryhodnou 32bitovou aplikaci, bude snížena úroveň na standardní vázané připojení.

Sdílená připojení (nezávislá na podprocesech) s MQCONNX

Tyto informace použijte k seznámení se se sdílenými připojeními s rozhraním MQCONNX a s některými poznámkami k použití, které je třeba zvážit.

Poznámka: Nepodporováno na produktu WebSphere MQ pro z/OS.

Na platformách WebSphere MQ jiných než WebSphere MQ for z/OSje připojení vytvořené pomocí produktu MQCONN dostupné pouze pro podproces, který vytvořil připojení. Volby v rámci volání MQCONNX umožňují vytvořit připojení, které může být sdíleno všemi podprocesy v procesu. Je-li aplikace spuštěna v transakčním prostředí, které vyžaduje zadání volání MQI ve stejném podprocesu, je třeba použít následující výchozí volbu:

MQCNO_HANDLE_SHARE_NONE
Vytvoří nesdílené připojení.

Ve většině ostatních prostředí můžete použít jednu z následujících možností sdíleného připojení podprocesů:

MQCNO_HANDLE_SHARE_BLOCK
Vytvoří sdílené připojení. Je-li v připojení k produktu MQCNO_HANDLE_SHARE_BLOCK aktuálně používáno připojení prostřednictvím volání MQI v jiném podprocesu, bude volání MQI čekat, dokud nebude dokončeno aktuální volání MQI.

MQCNO_HANDLE_SHARE_NO_BLOCK
Vytvoří sdílené připojení. Je-li v připojení k produktu MQCNO_HANDLE_SHARE_NO_BLOCK připojení aktuálně používáno voláním MQI v jiném podprocesu, volání MQI se okamžitě nezdaří s příčinou MQRC_CALL_IN_PROGRESS.

S výjimkou prostředí MTS (Microsoft Transaction Server) je výchozí hodnota MQCNO_HANDLE_SHARE_NONE. V prostředí MTS je předvolená hodnota MQCNO_HANDLE_SHARE_BLOCK.

Popisovač připojení je vrácen z volání MQCONNX . Popisovač může být použit následujícími voláními MQI z libovolného podprocesu v procesu, přidružování těchto volání k obslužné rutiny vrácené z produktu MQCONNX. Volání MQI s použitím jednoho sdíleného manipulátoru jsou serializována mezi podprocesy.

Například následující posloupnost aktivit je možná se sdíleným hantem:

1. Vlákno 1 vydává MQCONNX a získává sdílený popisovač *h1*
2. Podproces 1 otevře frontu a vydá požadavek na získání pomocí *h1*
3. Podproces 2 vydá požadavek na vložení pomocí *h1*
4. Podproces 3 vydá požadavek na vložení pomocí *h1*
5. Problémy podprocesů 2 MQDISC s použitím *h1*

Zatímco manipulátor je používán libovolným vláknem, přístup k připojení není k dispozici pro ostatní podprocesy. Za okolností, kdy je přijatelné, aby podproces čekal na dokončení nějaké předchozí volání z jiného podprocesu, použijte příkaz MQCONNX s volbou MQCNO_HANDLE_SHARE_BLOCK.

Blokování však může způsobit potíže. Předpokládejme, že v kroku “2” na stránce 204odešle podproces 1 požadavek na získání, který čeká na zprávy, které možná ještě nedorazily (get with wait). V tomto případě zůstanou podprocesy 2 a 3 také čekající (blokované) tak dlouho, jak dlouho trvá požadavek na získání požadavku na vlákno 1. Dáváte-li přednost tomu, aby se volání MQI vrátilo s chybou, pokud již bylo na manipulátoru spuštěno jiné volání MQI, použijte příkaz MQCONNX s volbou MQCNO_HANDLE_SHARE_NO_BLOCK.

Poznámky k použití sdíleného připojení

1. Všechny manipulátory objektů (Hobj) vytvořené otevřením objektu jsou přidruženy k Hconn; takže pro sdílený Hconn jsou objekty Hobjs také sdíleny a použitelné libovolným vláknem pomocí Hconn. Podobně je každá jednotka práce spouštěná pod Hconn asociována s tímto Hconn, takže je tento atribut sdílen všemi podprocesy se sdíleným Hconn.
2. *Jakýkoli* podproces může volat MQDISC k odpojení sdíleného připojení Hconn, nikoli pouze podprocesu, který volal odpovídající MQCONNX. Operace MQDISC ukončí modul Hconn, který ji znepřístupní pro všechny podprocesy.
3. Jeden podproces může použít více sdílených položek Hcons sériově, například pomocí příkazu MQPUT můžete vložit jednu zprávu pod jedním sdíleným Hconn a pak vložit jinou zprávu pomocí jiného sdíleného Hconn, přičemž každá operace bude pod jinou lokální pracovní jednotkou.
4. Sdílené HConns nelze použít v rámci globální transakce.

Použití voleb volání MQCONNX s MQ_CONNECT_TYPE

Pomocí těchto informací můžete porozumět různým volbám volání MQCONNX, jak jsou použity s MQ_CONNECT_TYPE.

V systémech WebSphere MQ for IBM i, WebSphere MQ for Windowsa WebSphere MQ v systémech UNIX and Linux můžete použít proměnnou prostředí MQ_CONNECT_TYPE v kombinaci s typem vazby zadaným v poli *Options* struktury MQCNO použité ve volání MQCONNX.

Tabulka 33. Proměnná 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.

Odpojení programů od správce front pomocí příkazu MQDISC

Tato část obsahuje informace o odpojování programů od správce front pomocí funkce MQDISC.

Když program, který se připojil ke správci front pomocí volání MQCONN nebo MQCONNX, dokončil veškerou interakci se správcem front, přeruší spojení pomocí volání MQDISC s výjimkou následujících:

- Na serveru CICS Transaction Server for z/OS , kde je volání volitelné, pokud nebylo použito MQCONNX a chcete zrušit značku připojení před ukončením aplikace.
- V produktu WebSphere MQ for IBM i , kde se při odhlášení z operačního systému provede implicitní volání MQDISC.

Jako vstup do volání MQDISC je třeba při připojení ke správci front dodat manipulátor připojení (Hconn), který byl vrácen příkazem MQCONN nebo MQCONNX.

S výjimkou systému CICS v systému z/OSpo volání funkce MQDISC již není manipulátor připojení (Hconn) nadále platný a nelze znovu vydat žádná další volání MQI, dokud znovu nezavoláte MQCONN nebo MQCONNX. MQDISC provádí implicitní MQCLOSE pro všechny objekty, které jsou stále otevřené pomocí tohoto ovladače.

Pokud použijete MQCONNX k připojení k produktu WebSphere MQ pro z/OS, MQDISC také ukončí rozsah značky připojení zavedené rozhraním MQCONNX. Pokud však v aplikaci CICS, IMSnebo aplikaci RRS existuje aktivní jednotka zotavení přidružená ke značce připojení, je hodnota MQDISC odmítnuta s kódem příčiny MQRC_CONN_TAG_NOT_RELEASED.

Popisy parametrů jsou uvedeny v popisu volání MQDISC v MQDISC.

Když není vydáno MQDISC

Při ukončení vytvářeného podprocesu se vyčistí standardní nesdílené připojení (Hconn). Sdílené připojení je implicitně odpojeno a odpojeno až po ukončení celého procesu. Pokud se podproces, který vytvořil sdílený Hconn, ukončí, zatímco Hconn stále existuje, je Hconn stále použitelný.

Kontrola oprávnění

Volání MQCLOSE a MQDISC obvykle neprovádí kontrolu oprávnění.

V normálním průběhu událostí se úloha, která má oprávnění k otevření nebo připojení k objektu WebSphere MQ , uzavře nebo odpojí od tohoto objektu. I když je oprávnění úlohy, která se připojila k objektu WebSphere MQ nebo je otevírány, přijata, volání MQCLOSE a MQDISC jsou akceptována.

Otevírání a zavírání objektů

Tyto informace poskytují náhled na otevření a zavření objektů produktu WebSphere MQ .

Chcete-li provést některou z následujících operací, je třeba nejprve *otevřít* příslušný objekt produktu WebSphere MQ :

- Vložení zpráv do fronty
- Získat (procházet nebo načíst) zprávy z fronty
- Nastavit atributy objektu
- Dotaz na atributy libovolného objektu

Použijte volání MQOPEN k otevření objektu pomocí voleb volání, abyste uvedli, co chcete dělat s objektem. Jedinou výjimkou je, že chcete-li vložit jednu zprávu do fronty, a ihned zavřít frontu. V takovém případě můžete vynechat fázi *otevírání* pomocí volání MQPUT1 (viz “Vložení jedné zprávy do fronty pomocí volání MQPUT1” na stránce 224).

Před otevřením objektu pomocí volání MQOPEN je třeba program připojit ke správci front. To je podrobně vysvětleno pro všechna prostředí v produktu “Připojování k správci front a odpojování od něj” na stránce 198.

K dispozici jsou čtyři typy objektů WebSphere MQ , které lze otevřít:

- Fronta
- Seznam názvů
- Definice procesu
- Správce front

Všechny tyto objekty otevírejte podobným způsobem pomocí volání MQOPEN. Další informace o objektech produktu WebSphere MQ naleznete v tématu Objekty.

Stejný objekt můžete otevřít více než jednou, a pokaždé, když získáte nový popisovač objektu.Možná budete chtít procházet zprávy ve frontě pomocí jednoho úchytu a odebrat zprávy ze stejné fronty s použitím jiného ovladače. To šetří využití prostředků k uzavření a opětnému otevření téhož objektu. Můžete také otevřít frontu pro procházení *a* odebíráním zpráv současně.

Kromě toho můžete otevřít více objektů s jedinou operací MQOPEN a zavřít je pomocí příkazu MQCLOSE. Informace o tom, jak to provést, viz “Distribuční seznamy” na stránce 226 .

Při pokusu o otevření objektu správce front zkontroluje, zda máte autorizaci k otevření daného objektu pro volby, které jste zadali v rámci volání MQOPEN.

Objekty jsou automaticky zavřeny, když se program odpojí od správce front. V prostředí IMS je odpojení vynuceno, když program zahájí zpracování pro nového uživatele následujícího po volání GU (get unique) IMS . Na platformě IBM i jsou objekty automaticky zavřeny při ukončení úlohy.

Je dobrým programovacím zvykem zavřít objekty, které jste otevřeli. Použijte volání MQCLOSE k provedení tohoto.

Chcete-li zjistit více o otevírání a zavírání objektů, použijte následující odkazy:

- “Otevírání objektů pomocí volání MQOPEN” na stránce 207
- “Vytváření dynamických front” na stránce 215
- “Otevírání vzdálených front” na stránce 215
- “Zavírání objektů pomocí volání MQCLOSE” na stránce 216

Související pojmy

“Přehled rozhraní fronty zpráv” na stránce 187
Zde jsou uvedeny informace o komponentách rozhraní MQI (Message Queue Interface).

“Připojování k správci front a odpojování od něj” na stránce 198
Chcete-li používat programovací služby produktu WebSphere MQ , musí mít program připojení ke správci front. V této části se dozvíte, jak se připojit k správci front a odpojit je od správce front.

“Vložení zpráv do fronty” na stránce 216
V této části se dozvíte, jak vložit zprávy do fronty.

“Získávání zpráv z fronty” na stránce 231
Tyto informace použijte k získání informací o získávání zpráv z fronty.

“Inquaning about a nastavení atributů objektu” na stránce 307
Atributy jsou vlastnosti, které definují charakteristiky objektu produktu WebSphere MQ .

“Potvrzení a zálohování jednotek práce” na stránce 310
Tyto informace popisují, jak potvrdit a zazálohovat všechny zotavitelné operace get a put, které se vyskytly v jednotce práce.

“Spuštění aplikací produktu IBM WebSphere MQ pomocí spouštěčů” na stránce 316
Informace o spouštěčích a o tom, jak spustit aplikace IBM WebSphere MQ pomocí spouštěčů.

“Práce s MQI a klastry” na stránce 332
Existují speciální volby na voláních a návratových kódech, které se vztahují ke klastrování.

Otevírání objektů pomocí volání MQOPEN

Tyto informace použijte k seznámení se s otevíráním objektů pomocí volání MQOPEN.

Vstup do volání MQOPEN je třeba zadat:

- Popisovač připojení. Pro aplikace 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 obslužnou rutinu připojení vrácenou voláním MQCONN nebo MQCONNX.
- Popis objektu, který chcete otevřít, s použitím struktury deskriptoru objektu (MQOD).
- Jedna nebo více voleb, které řídí akci volání.

Výstup z MQOPEN je:

- Popisovač objektu, který představuje váš přístup k objektu. Tato hodnota se používá při zadávání veškerých následných volání MQI.

- Upravená struktura popisovače objektu, pokud vytváříte dynamickou frontu (a je podporována na vaší platformě).
- Kód dokončení.
- Kód příčiny.

Rozsah manipulátoru objektu

Rozsah manipulátoru objektu (Hobj) je stejný jako rozsah platnosti manipulátoru připojení (Hconn).

To je zahrnuto v "Rozsah MQCONN nebo MQCONNX" na stránce 200 a "Sdílená připojení (nezávislá na podprocesech) s MQCONNX" na stránce 204. V některých prostředích však existují další pokyny:

CICS
V programu CICS můžete použít popisovač pouze v rámci stejné úlohy CICS , ze které jste provedli volání MQOPEN.

Dávkové zpracování IMS a z/OS
V prostředí IMS a v dávkových prostředích můžete použít popisovač v rámci stejné úlohy, ale ne v rámci žádných dílčích úloh.

Popisy parametrů volání MQOPEN jsou uvedeny v části MQOPEN.

Následující oddíly popisují informace, které je třeba dodat jako vstup pro funkci MQOPEN.

Označení objektů (struktura MQOD)

K identifikaci objektu, který chcete otevřít, použijte strukturu MQOD. Tato struktura je vstupním parametrem pro volání MQOPEN. (Strukturu upraví správce front, když použijete volání MQOPEN k vytvoření dynamické fronty.)

Podrobné informace o struktuře MQOD naleznete v části MQOD.

Informace o použití struktury MQOD pro distribuční seznamy naleznete v části "Použití struktury MQOD" na stránce 227 v části "Distribuční seznamy" na stránce 226.

Rozpoznání názvu

Jak volání MQOPEN řeší názvy front a správců front.

Poznámka: Alias správce front je definice vzdálené fronty bez pole RNAME .

Při otevření fronty produktu WebSphere MQ provede volání MQOPEN funkci rozpoznání názvu ve vámi specifiutě názvu fronty. To určuje, jakou frontu provádí správce front následné operace. To znamená, že když uvedete název alias fronty nebo vzdálené fronty v deskriptoru objektu (MQOD), volání vyřeší název buď do lokální fronty, nebo do přenosové fronty. Je-li fronta otevřena pro libovolný typ vstupu, procházení nebo nastavení, interpretuje se jako lokální fronta, pokud existuje, a selže, pokud tam není. Překládá se na nelokální frontu pouze tehdy, je-li otevřena pouze pro výstup, dotazuje se pouze nebo pouze na výstup a dotazuje se. Přehled procesu rozlišování názvů najdete v tématu Tabulka 34 na stránce 209 . Název, který jste zadali v produktu *ObjectQMgrName* , je vyřešen *před* , než je uvedeno v *ObjectName*.

Tabulka 34 na stránce 209 také ukazuje, jak můžete použít lokální definici vzdálené fronty k definování aliasu pro název správce front. To vám umožní určit, která přenosová fronta se použije při vkládání zpráv do vzdálené fronty, takže byste mohli například použít jednu přenosovou frontu pro zprávy určené pro mnoho vzdálených správců front.

Chcete-li použít následující tabulku, nejprve si přečtěte dva levé sloupce pod záhlavím **Vstup do MQOD**a vyberte příslušný případ. Poté si přečtěte všechny pokyny v příslušném řádku. Podle pokynů ve sloupcích **Vyřešené názvy** se můžete vrátit buď do sloupců **Vstup do MQOD** a vložit hodnoty podle pokynů, nebo můžete tabulku ukončit s dodanými výsledky. Můžete být například vyzváni k zadání *ObjectName*.

Tabulka 34. Řešení názvů front při použití funkce MQOPEN

Vstup pro MQOD		**Převedené názvy**		
ObjectQMgrName	***ObjectName***	***ObjectQMgrName***	***ObjectName***	**Přenosová fronta**
Prázdný nebo lokální správce front	Lokální fronta bez atributu CLUSTER	Lokální správce front	Zadejte *ObjectName*	Nehodí se (lokální používaná fronta)
Prázdný správce front	Lokální fronta s atributem CLUSTER	Vybraný správce front klastru vybraného klastru správy pracovní zátěže nebo specifický správce front klastru vybraný na operaci PUT	Zadejte *ObjectName*	SYSTEM.CLUSTER.TRANSMIT.QUEUE a použitá lokální fronta SYSTEM.QSG.TRANSMIT.QUEUE (viz poznámka)
Lokální správce front	Lokální fronta s atributem CLUSTER	Lokální správce front	Zadejte *ObjectName*	Nehodí se (lokální používaná fronta)
Prázdný nebo lokální správce front	Modelová fronta	Lokální správce front	Generovaný název	Nehodí se (lokální používaná fronta)
Prázdný nebo lokální správce front	Alias fronty s atributem CLUSTER nebo bez ní	Znovu proveďte interpretaci názvu s názvem *ObjectQMgrNázev* a zadejte vstupní hodnotu *ObjectName* do hodnoty *BaseQName* v objektu definice alias fronty. Nesmí se interpretovat jako lokálně definovaný alias, kde je zadán *ObjectQMgrNázev* , ale může se převést na klastrovaný alias (hostovaný na jiných správcích front), kde je *ObjectQMgrNázev* prázdný.		
Lokální správce front	Alias fronty s atributem CLUSTER	Alias nesmí být převáděn na frontu klastru, která není lokálně definována, nebo pro frontu klastru obsahující stejnou hodnotu *ObjectName* jako alias.		
Prázdný správce front	Alias fronty s atributem CLUSTER	Alias se může interpretovat jako fronta klastru se stejnou hodnotou *ObjectName* jako alias.		

Tabulka 34. Řešení názvů front při použití funkce MQOPEN (pokračování)

Vstup pro MQOD		**Převedené názvy**		
ObjectQMgrName	***ObjectName***	***ObjectQMgrName***	***ObjectName***	**Přenosová fronta**
Prázdný nebo lokální správce front	lokální definice vzdálené fronty	Znovu proveďte řešení názvu s parametrem *ObjectQMgrName* nastaveným na hodnotu *RemoteQMgrName*a *Obj ectName* nastavenou na *RemoteQName*. Nesmí se interpretovat vzdálené fronty		Název atributu *XmitQName* , je-li jiný než blank; v opačném případě *RemoteQMgrName* v objektu definice vzdálené fronty. SYSTEM.QSG.TRANSMI T.QUEUE (viz poznámka)
Prázdný správce front	Nebyl nalezen odpovídající lokální objekt; byla nalezena fronta klastru	Vybraný správce front klastru vybraného klastru správy pracovní zátěže nebo specifický správce front klastru vybraný na operaci PUT	Zadejte *ObjectName*	SYSTEM.CLUSTER.TRAN SMIT.QUEUE SYSTEM.QSG.TRANSMI T.QUEUE (viz poznámka)
Prázdný nebo lokální správce front	Nebyl nalezen odpovídající lokální objekt; fronta klastru nebyla nalezena		Chyba, fronta nebyla nalezena	Nelze použít
Název správce front ve stejné skupině sdílení front jako lokální správce front	Lokální sdílená fronta	Lokální správce front	Zadejte *ObjectName*	Nelze použít
Název lokální přenosové fronty	(Nevyřešeno)	Zadejte *ObjectQMgrNázev*	Zadejte *ObjectName*	Zadejte *ObjectQMgrNázev* SYSTEM.QSG.TRANSMI T.QUEUE (viz poznámka)
Definice alias správce front (*RemoteQMgrName* může být lokální správce front)	(Nevyřešeno, vzdálená fronta)	Znovu proveďte řešení názvu s parametrem *ObjectQMgrName* nastaveným na hodnotu *RemoteQMgrName*. Musí se interpretovat na vzdálené fronty	Zadejte *ObjectName*	Název atributu *XmitQName* , je-li jiný než blank; v opačném případě *RemoteQMgrName* v objektu definice vzdálené fronty. SYSTEM.QSG.TRANSMI T.QUEUE (viz poznámka)

Tabulka 34. Řešení názvů front při použití funkce MQOPEN (pokračování)				
Vstup pro MQOD		**Převedené názvy**		
ObjectQMgrName	***ObjectName***	***ObjectQMgrName***	***ObjectName***	**Přenosová fronta**
Správce front není názvem žádného lokálního objektu; byl nalezen správce front klastru nebo alias správce front.	(Nevyřešeno)	*ObjectQMgrNázev* nebo specifický správce front klastru vybraný na PUT	Zadejte *ObjectName*	SYSTEM.CLUSTER.TRANSMIT.QUEUE SYSTEM.QSG.TRANSMIT.QUEUE (viz poznámka)
Správce front není názvem žádného lokálního objektu; nebyly nalezeny žádné objekty klastru	(Nevyřešeno)	Zadejte *ObjectQMgrNázev*	Zadejte *ObjectName*	Atribut *DefXmitQName* správce front, ve kterém je podporován *DefXmitQName* . SYSTEM.QSG.TRANSMIT.QUEUE (viz poznámka)

Notes:

1. *BaseQName* je název základní fronty z definice alias fronty.
2. *RemoteQName* je název vzdálené fronty z lokální definice vzdálené fronty.
3. *RemoteQMgrName* je název vzdáleného správce front z lokální definice vzdálené fronty.
4. *XmitQName* je název přenosové fronty z lokální definice vzdálené fronty.
5. Při použití produktu WebSphere MQ pro správce front systému z/OS , kteří jsou součástí skupiny sdílení front (QSG), lze místo názvu lokálního správce front v produktu Tabulka 34 na stránce 209použít název skupiny QSG.

   Pokud lokální správce front nemůže otevřít cílovou frontu nebo vložit do fronty zprávu, bude zpráva přenesena do určeného názvu ObjectQMgrbuď prostřednictvím fronty v rámci skupiny, nebo kanálu produktu WebSphere MQ .
6. Ve sloupci *ObjectName* tabulky se KLASTR odkazuje na atributy CLUSTER a CLUSNL fronty.
7. SYSTEM.QSG.TRANSMIT.QUEUE se používá v případě, že se lokální a vzdálené správci front nacházejí ve stejné skupině sdílení front; řazení do fronty v rámci skupiny je povoleno.
8. Pokud jste každému klastru odesílacímu kanálu přiřadili jinou přenosovou frontu klastru, SYSTEM.CLUSTER.TRANSMIT.QUEUE nemusí být název přenosové fronty klastru. Další informace o více přenosových frontách klastru najdete v tématu Klastrování: Plánování konfigurace přenosových front klastru .
9. V situaci, kdy správce front není názvem žádného lokálního objektu, správce front klastru nebo alias správce front, který byl nalezen.

   Pokud jste zadali název správce front pomocí produktu **ObjectQMgrName**a existuje více kanálů klastru s různými názvy klastrů, které jsou známy lokálním správcem front, které by dosáhly místa určení, může být kterýkoli z těchto kanálů použit k přesunutí zprávy bez ohledu na název klastru cílové fronty.

   To může být neočekávané, pokud jste očekávali zprávy pro tuto frontu pouze k odeslání prostřednictvím kanálu se stejným názvem klastru jako fronta.

   Produkt **ObjectQMgrName** však v tomto případě má přednost a vyrovnávání pracovní zátěže klastru bere v úvahu všechny kanály, které se mohou dostat k tomuto správci front, bez ohledu na název klastru, ve kterém se nacházejí.

Při otevření fronty aliasu se také otevře základní fronta, do níž je alias převeden, a při otevření vzdálené fronty se také otevře přenosová fronta. Proto nemůžete odstranit frontu, kterou jste určili, nebo frontu, na kterou se rozlišuje, zatímco druhá je otevřená.

Fronta aliasů se sice nemůže přeložit na jinou lokálně definovanou alias frontu (sdílenou v klustru nebo ne), takže je možné ji přeložit na vzdáleně definovanou frontu aliasů klastru, a proto může být zadána jako základní fronta.

Vyřešený název fronty a vyřešený název správce front jsou uloženy v polích *ResolvedQName* a *ResolvedQMgrName* v MQOD.

Další informace o rozlišení názvu v distribuovaném prostředí fronty najdete v tématu Co je to rozlišení názvu fronty?.

Použití voleb volání MQOPEN

V parametru *Options* volání MQOPEN musíte vybrat jednu nebo více voleb pro řízení přístupu, který jste dostali k objektu, který otevíráte. S těmito volbami můžete:

- Otevřete frontu a určete, že všechny zprávy zařazené do této fronty musí být směrovány do stejné instance.
- Chcete-li povolit vkládání zpráv do fronty, otevřete ji.
- Otevřete frontu, abyste mohli procházet zprávy na této frontě
- Otevřete frontu, abyste v něm mohli odebírat zprávy
- Otevřít objekt, který vám umožní dotazovat se na atributy a nastavit jeho atributy (ale můžete nastavit pouze atributy front)
- Otevřít téma nebo řetězec tématu pro publikování zpráv v rámci tohoto tématu
- Přidružit informace o kontextu ke zprávě
- Nominovat alternativní identifikátor uživatele, který má být použit pro kontroly zabezpečení
- Řídit volání, je-li správce front ve stavu uvedení do klidového stavu

Volba MQOPEN pro frontu klastru
Vazba použitá pro 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 333.

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 vložení zpráv
Chcete-li otevřít frontu nebo téma pro vložení zpráv do fronty, použijte volbu MQOO_OUTPUT.

Volba MQOPEN pro procházení zpráv
Chcete-li otevřít frontu, aby bylo možné *procházet* zprávy, použijte volání MQOPEN s volbou MQOO_BROWSE.

Tím se vytvoří *kurzor procházení* , který správce front používá k identifikaci další zprávy ve frontě. Další informace naleznete v části “Procházení zpráv ve frontě” na stránce 260.

Poznámka:

1. Zprávy ve vzdálené frontě nelze procházet, neotvírejte vzdálenou frontu pomocí volby MQOO_BROWSE.
2. Při otevírání rozdělovníku nelze tuto volbu uvést. Další informace o distribučních seznamech viz “Distribuční seznamy” na stránce 226.
3. Funkci MQOO_CO_OP spolu s aplikací MQOO_BROWSE použijte v případě, že používáte spolupracující procházení, viz Volby .

Volby MQOPEN pro odebrání zpráv
K odebrání zpráv z fronty slouží tři volby řízení otevírání fronty.

V libovolném volání MQOPEN můžete použít pouze jeden z nich. Tyto volby definují, zda má váš program výhradní nebo sdílený přístup do fronty. *Výlučný přístup* znamená, že dokud nezavřete frontu,z ní můžete odebírat pouze zprávy. Pokud se jiný program pokusí otevřít frontu za účelem odebrání zpráv, volání MQOPEN se nezdaří. *Sdílený přístup* znamená, že více než jeden program může odebratz fronty.

Nejvhodnější metodou je přijetí typu přístupu, který byl určen pro frontu, když byla fronta definována. Definice fronty zahrnuje nastavení *Shareability* aAtributy produktu *DefInputOpenOption* . Chcete-li tento přístup přijmout, použijte volbu MQOO_INPUT_AS_Q_DEF. Podívejte se na Tabulka 35 na stránce 213 , abyste zjistili, jak nastavení těchto atributů ovlivňuje typ přístupu, který bude poskytnut při použití této volby.

Tabulka 35. Jak atributy fronty a volby volání MQOPEN mají vliv na přístup k frontám

Atributy fronty		**Typ přístupu s volbami MQOPEN**		
Shareability	*DefInputOpenOption*	**AS Q_DEF**	**SHARED**	**EXCLUSIVE**
Možnost sdílení	SHARED	sdíleno	sdíleno	výlučný
Možnost sdílení	EXCLUSIVE	výlučný	sdíleno	výlučný
NESDÍLET *	SDÍLENO *	výlučný	výlučný	výlučný
NESDÍLET	EXCLUSIVE	výlučný	výlučný	výlučný
Poznámka: * Ačkoli můžete definovat frontu pro použití této kombinace atributů, je výchozí vstupní otevřená volba potlačena atributem sdílené schopnosti.				

Alternativně:

- Pokud víte, že vaše aplikace může úspěšně pracovat i v případě, že jiné programy mohou zprávy z fronty odstranit současně, použijte volbu MQOO_INPUT_SHARED. Tabulka 35 na stránce 213 ukazuje, jak v některých případech budete mít výlučný přístup do fronty, a to i s touto volbou.
- Pokud víte, že vaše aplikace může fungovat úspěšně pouze v případě, že jiným programům není zabráněno v odstraňování zpráv z fronty současně, použijte volbu MQOO_INPUT_EXCLUSIVE.

Poznámka:

1. Zprávy ze vzdálené fronty nelze odebrat. Proto nelze vzdálenou frontu otevřít pomocí žádné z voleb MQOO_INPUT_ *.
2. Při otevírání rozdělovníku nelze tuto volbu uvést. Další informace uvádí téma “Distribuční seznamy” na stránce 226.

Volby MQOPEN pro nastavení a zjišťování informací o atributech
Chcete-li otevřít frontu tak, abyste mohli nastavit její atributy, použijte volbu MQOO_SET.

Atributy žádného jiného typu objektu nelze nastavit (viz “Inquaning about a nastavení atributů objektu” na stránce 307).

Chcete-li otevřít objekt tak, abyste se mohli dotázat na jeho atributy, použijte volbu MQOO_INQUIRE.

Poznámka: Při otevírání rozdělovníku nelze tuto volbu uvést.

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

Volby vám umožňují rozlišovat mezi informacemi o kontextu, které souvisí s *uživatelem* , který je původcem zprávy, a to, které se vztahuje k *aplikaci* , která je původcem zprávy. Můžete také zvolit nastavení kontextové informace při vložení zprávy do fronty nebo můžete zvolit přepnutí kontextu automaticky z jiného manipulátorů fronty.

Související pojmy
“kontext zprávy” na stránce 37
Informace *Kontext zprávy* umožňují aplikaci, která načte zprávu, zjistit informace o odesílateli zprávy.

“Řízení informací o kontextu” na stránce 222
Použijete-li volání MQPUT nebo MQPUT1 k vložení zprávy do fronty, můžete určit, že správce front má přidat některé výchozí kontextové informace do deskriptoru zpráv. Aplikace, které mají odpovídající úroveň oprávnění, mohou přidávat další informace o kontextu. Pole voleb ve struktuře MQPMO můžete použít k řízení informací o kontextu.

Volba MQOPEN pro alternativní oprávnění uživatele
Pokusíte-li se otevřít objekt pomocí volání MQOPEN, správce front zkontroluje, zda máte oprávnění k otevření daného objektu. Pokud nemáte oprávnění, volání selže.

Serverová programy však mohou chtít správce front zkontrolovat autorizaci uživatele, pro kterého pracují, spíše než vlastní autorizace serveru. K tomu musí použít volbu MQOO_ALTERNATE_USER_AUTHORITY pro volání MQOPEN a určit alternativní ID uživatele v poli *AlternateUserId* struktury MQOD. Obvykle server získá ID uživatele z informací o kontextu ve zprávě, kterou zpracovává.

Volba MQOPEN pro uvedení správce front do klidového stavu
Pokud v prostředí CICS v systému z/OSpoužijete volání MQOPEN, je-li správce front ve stavu uvedení do klidového stavu, volání vždy selže.

V jiných prostředích z/OS , IBM i, Windows a v prostředí systémů UNIX and Linux se volání nezdaří, je-li správce front uváděn do klidového stavu pouze tehdy, když použijete volbu MQOO_FAIL_IF_QUIESCING volání MQOPEN.

Volba MQOPEN pro interpretaci názvů lokálních front
Když otevřete lokální, alias nebo modelovou frontu, vrátí se lokální fronta.

Když však otevřete vzdálenou frontu nebo frontu klastru, pole *ResolvedQName* a *ResolvedQMgrName* struktury MQOD jsou vyplněny názvy vzdálené fronty a vzdáleného správce front nalezeného v definici vzdálené fronty nebo ve zvolené vzdálené frontě klastru.

Použijte volbu MQOO_RESOLVE_LOCAL_Q volání MQOPEN k vyplnění hodnoty *ResolvedQName* ve struktuře MQOD s názvem lokální fronty, která byla otevřena. *ResolvedQMgrName* je podobně zaplněn názvem lokálního správce front, který je hostitelem lokální fronty. Toto pole je dostupné pouze u verze 3 struktury MQOD; je-li struktura menší než verze 3, je MQOO_RESOLVE_LOCAL_Q ignorována, aniž by byla vrácena chyba.

Určíte-li hodnotu MQOO_RESOLVE_LOCAL_Q, je-li například otevřena vzdálená fronta, *ResolvedQName* je název přenosové fronty, do které budou zprávy vloženy. *ResolvedQMgrName*je název lokálního správce front, který je hostitelem přenosové fronty.

Vytváření dynamických front

Dynamickou frontu použijte, když nepotřebujete frontu po ukončení aplikace.

Můžete například použít dynamickou frontu pro svou frontu pro odpověď. Název fronty pro odpověď zadejte do pole *ReplyToQ* struktury MQMD při vložení zprávy do fronty (viz “Definování zpráv pomocí struktury MQMD” na stránce 218).

Chcete-li vytvořit dynamickou frontu, použijte šablonu známou jako modelovou frontu spolu s voláním MQOPEN. Modelovou frontu vytvoříte pomocí příkazů produktu WebSphere MQ nebo z panelů operací a ovládacích panelů. Dynamická fronta, kterou vytvoříte, převezme atributy modelové fronty.

Když voláte MQOPEN, zadejte název modelové fronty do pole *ObjectName* struktury MQOD. Po dokončení volání je pole *ObjectName* nastaveno na název dynamické fronty, která je vytvořena. Pole *ObjectQMgrName* je také nastaveno na název lokálního správce front.

Název dynamické fronty, kterou vytvoříte, můžete zadat třemi způsoby:

- Zadejte celé jméno, které chcete v poli *DynamicQName* struktury MQOD.
- Uveďte předponu (méně než 33 znaků) pro název a umožněte správci front generovat zbývající část názvu. To znamená, že správce front vygeneruje jedinečný název, ale stále máte nějaký ovládací prvek (například můžete chtít, aby každý uživatel používal určitou předponu, nebo můžete chtít udělit speciální klasifikaci zabezpečení pro fronty s určitou předponou v jejich jménu). Chcete-li použít tuto metodu, uveďte hvězdičku (*) pro poslední neprázdný znak pole *DynamicQName* . Neuvádějte pro název dynamické fronty jednu hvězdičku (*).
- Povolit správci front generovat úplný název. Chcete-li použít tuto metodu, uveďte hvězdičku (*) v první znakové pozici pole *DynamicQName* .

Další informace o těchto metodách naleznete v popisu pole DynamicQName .

Existují další informace o dynamických frontách v Dynamické a modelové frontě .

Otevírání vzdálených front

Vzdálená fronta je fronta, jejímž vlastníkem je jiný správce front, než ten, ke kterému je aplikace připojena.

Chcete-li otevřít vzdálenou frontu, použijte volání MQOPEN jako pro lokální frontu. Název fronty můžete zadat následujícím způsobem:

1. V poli *ObjectName* struktury MQOD zadejte název vzdálené fronty, jak je známo, do správce front *local* .

   Poznámka: V tomto případě ponechte pole *ObjectQMgrName* prázdné.
2. V poli *ObjectName* struktury MQOD zadejte název vzdálené fronty, jak je známo ve správci front *remote* . Do pole *ObjectQMgrName* zadejte:

   - Název přenosové fronty, která má stejný název jako vzdálený správce front. Název a velikost písmen (velká písmena, malá písmena nebo směs) se musí přesně shodovat s *přesně*.
   - Název objektu aliasu správce front, který je interpretovaný jako správce cílové fronty nebo pro přenosovou frontu.

   To sděluje správci front místo určení zprávy a také přenosovou frontu, kterou je třeba pro získání této zprávy umístit.
3. Je-li v poli *ObjectName* struktury MQOD podporováno *DefXmitQname* , zadejte název vzdálené fronty, jak je znám ve *vzdáleném* správci front.

   Poznámka: Nastavte pole *ObjectQMgrName* na název vzdáleného správce front (v tomto případě nemůže být ponecháno prázdné).

Při volání MQOPEN jsou ověřovány pouze lokální názvy. Poslední kontrola existence přenosové fronty, která má být použita.

Tyto metody jsou shrnuty v Tabulka 34 na stránce 209.

Zavírání objektů pomocí volání MQCLOSE

Chcete-li objekt zavřít, použijte volání MQCLOSE.

Je-li objektem fronta, povšimněte si následujících:

- Před zavřením není nutné vyprázdnit dočasnou dynamickou frontu.

  Když zavřete dočasnou dynamickou frontu, odstraní se fronta spolu se všemi zprávami, které se na ní mohou stále nacházet. To platí i v případě, že existují nepotvrzené příkazy MQGET, MQPUT nebo MQPUT1 nevyřízené pro danou frontu.
- Pokud jste v produktu WebSphere MQ for z/OSzrušili všechny požadavky MQGET s nevyřízenou volbou MQGMO_SET_SIGNAL pro tuto frontu, budou zrušeny.
- Pokud jste frontu otevřeli pomocí volby MQOO_BROWSE, je váš kurzor procházení zničen.

Uzavření nesouvisí se synchronizačním bodem, takže můžete zavřít fronty před nebo po bodu synchronizace.

Jako vstup do volání MQCLOSE je třeba dodat:

- Popisovač připojení. Použijte stejný manipulátor připojení k otevření nebo alternativně pro aplikace CICS v systému z/OS, můžete zadat konstantu MQHC_DEF_HCONN (která má hodnotu nula).
- Popisovač objektu, který chcete zavřít. Získejte jej z výstupu volání MQOPEN.
- MQCO_NONE v poli *Options* (pokud nezavíráte trvalou dynamickou frontu).
- Pomocí této volby lze určit, zda má správce front frontu odstranit i v případě, že v ní stále existují zprávy (při zavírání trvalé dynamické fronty).

Výstup z MQCLOSE je:

- Kód dokončení
- Kód příčiny
- Popisovač objektu, reset na hodnotu MQHO_UNUSABLE_HOTBJ

Popisy parametrů volání MQCLOSE jsou uvedeny v části MQCLOSE.

Vložení zpráv do fronty

V této části se dozvíte, jak vložit zprávy do fronty.

K vložení zpráv do fronty použijte volání MQPUT. Pomocí opakovaného volání MQPUT lze opakovaně vkládat zprávy do stejné fronty po počátečním volání MQOPEN. Až dokončíte vkládání všech zpráv do fronty, volejte MQCLOSE.

Chcete-li vložit jednu zprávu do fronty a zavřít frontu okamžitě, můžete použít volání MQPUT1 . Příkaz MQPUT1 provádí stejné funkce jako následující posloupnost volání:

- MQOPEN
- MQPUT
- MQCLOSE

Obecně však platí, že pokud máte více než jednu zprávu, která má být vložena do fronty, je efektivnější použít volání MQPUT. Závisí to na velikosti zprávy a na platformě, na které pracujete.

Chcete-li zjistit více o vkládání zpráv do fronty, použijte následující odkazy:

- “Vložení zpráv do lokální fronty pomocí volání MQPUT” na stránce 217
- “Vložení zpráv do vzdálené fronty” na stránce 222
- “Nastavení vlastností zprávy” na stránce 222
- “Řízení informací o kontextu” na stránce 222
- “Vložení jedné zprávy do fronty pomocí volání MQPUT1” na stránce 224

- “Distribuční seznamy” na stránce 226
- “Některé případy, kdy volání vložení selžou” na stránce 230

Související pojmy

“Přehled rozhraní fronty zpráv” na stránce 187
Zde jsou uvedeny informace o komponentách rozhraní MQI (Message Queue Interface).

“Připojování k správci front a odpojování od něj” na stránce 198
Chcete-li používat programovací služby produktu WebSphere MQ , musí mít program připojení ke správci front. V této části se dozvíte, jak se připojit k správci front a odpojit je od správce front.

“Otevírání a zavírání objektů” na stránce 206
Tyto informace poskytují náhled na otevření a zavření objektů produktu WebSphere MQ .

“Získávání zpráv z fronty” na stránce 231
Tyto informace použijte k získání informací o získávání zpráv z fronty.

“Inquaning about a nastavení atributů objektu” na stránce 307
Atributy jsou vlastnosti, které definují charakteristiky objektu produktu WebSphere MQ .

“Potvrzení a zálohování jednotek práce” na stránce 310
Tyto informace popisují, jak potvrdit a zazálohovat všechny zotavitelné operace get a put, které se vyskytly v jednotce práce.

“Spuštění aplikací produktu IBM WebSphere MQ pomocí spouštěčů” na stránce 316
Informace o spouštěčích a o tom, jak spustit aplikace IBM WebSphere MQ pomocí spouštěčů.

“Práce s MQI a klastry” na stránce 332
Existují speciální volby na voláních a návratových kódech, které se vztahují ke klastrování.

Vložení zpráv do lokální fronty pomocí volání MQPUT

Tyto informace použijte k získání informací o vkládání zpráv do lokální fronty pomocí volání MQPUT.

Jako vstup do volání MQPUT musíte zadat:

- Popisovač připojení (Hconn).
- Popisovač fronty (Hobj).
- Popis zprávy, kterou chcete vložit do fronty. To je ve formě struktury deskriptoru zpráv (MQMD).
- Řídicí informace ve formě struktury voleb put-message (MQPMO).
- Délka dat obsažených ve zprávě (MQLONG).
- Samotná data zprávy.

Výstup z volání MQPUT je následující:

- Kód příčiny (MQLONG)
- Kód dokončení (MQLONG)

Pokud je volání úspěšně dokončeno, vrací také strukturu vaší volby a strukturu deskriptoru zprávy. Volání upravuje strukturu voleb tak, aby zobrazovala název fronty a správce front, do kterého byla zpráva odeslána. Pokud požadujete, aby správce front generoval jedinečnou hodnotu pro identifikátor zprávy, kterou umisťujete (zadáním binární nuly v poli *MsgId* struktury MQMD), volání vloží hodnotu do pole *MsgId* před vrácením této struktury na vás. Tuto hodnotu obnovte, než vydáte další požadavek MQPUT.

V MQPUTje uveden popis volání MQPUT.

Další informace o informacích potřebných jako vstup pro volání MQPUT naleznete v následujících odkazech:

- “Určení manipulátorů” na stránce 218
- “Definování zpráv pomocí struktury MQMD” na stránce 218
- “Určení voleb pomocí struktury MQPMO” na stránce 218
- “Data ve zprávě” na stránce 221

- “Vložení zpráv: Použití obslužných rutin zpráv” na stránce 222

Určení manipulátorů

Pro popisovač připojení (*Hconn*) v systému CICS v aplikacích z/OS můžete zadat konstantu MQHC_DEFF_HCONN (má hodnotu nula) nebo můžete použít manipulátor připojení vrácený voláním MQCONN nebo MQCONNX. Pro ostatní aplikace vždy použijte obslužnou rutinu připojení vrácenou voláním MQCONN nebo MQCONNX.

V jakémkoli prostředí, ve kterém pracujete, použijte stejný popisovač fronty (*Hobj*), který je vrácen voláním MQOPEN.

Definování zpráv pomocí struktury MQMD

Struktura deskriptoru zpráv (MQMD) je vstupní/výstupní parametr pro volání MQPUT a MQPUT1 . Použijte jej k definování zprávy, kterou vkládáte do fronty.

Je-li MQPRI_PRIORITY_AS_Q_DEF nebo MQPER_PERSISTENCE_AS_Q_DEF zadán pro zprávu a fronta je fronta klastru, použijí se hodnoty fronty, na které se má provést příkaz MQPUT. Je-li tato fronta zakázána pro operaci MQPUT, volání se nezdaří. Další informace naleznete v tématu Konfigurace klastru správce front .

Poznámka: Použijte MQPMO_NEW_MSG_ID a MQPMO_NEW_CORREL_ID před vložením nové zprávy, abyste se ujistili, že jsou *MsgId* a *CorrelId* jedinečné. Hodnoty v těchto polích se vrátí v úspěšném volání MQPUT.

Existuje úvod do vlastností zprávy, které MQMD popisuje v produktu “Zprávy produktu IBM WebSphere MQ” na stránce 9, a v MQMDje popis struktury samotné.

Určení voleb pomocí struktury MQPMO

Použijte strukturu MQPMO (Vložit volbu zprávy) k předání voleb pro volání MQPUT a MQPUT1 .

Následující oddíly vám pomohou při vyplňování polí této struktury. Popis struktury v MQPMOje uveden v popisu struktury.

Struktura obsahuje následující pole:

- *StrucId*
- *Version*
- *Options*
- *Context*
- *ResolvedQName*
- *ResolvedQMgrName*
- *RecsPresent*
- *PutMsgRecsFields*
- *ResponseRecOffset and ResponseRecPtr*
- *OriginalMsgHandle*
- *NewMsgHandle*
- *Action*
- *PubLevel*

Obsah těchto polí je následující:

StrucId
Identifikuje strukturu jako strukturu voleb vložení zprávy. Jedná se o čtyřznakové pole. Vždy zadejte MQPMO_STRUC_ID.

Verze
Popisuje číslo verze struktury. Výchozí hodnota je MQPMO_VERSION_1. Zadáte-li MQPMO_VERSION_2, můžete použít distribuční seznamy (viz “Distribuční seznamy” na stránce 226). Zadáte-li MQPMO_VERSION_3, můžete použít popisovače zpráv a vlastnosti zprávy. Zadáte-li příkaz MQPMO_CURRENT_VERSION, bude vaše aplikace nastavena vždy tak, aby používala nejnovější úroveň.

Volby
Tento ovládací prvek řídí následující:

- Zda je operace vložení zahrnuta do pracovní jednotky
- Kolik kontextových informací je přidruženo ke zprávě
- Místo, odkud jsou informace o kontextu převzaty
- Zda se volání nezdaří, je-li správce front ve stavu uvedení do klidového stavu
- Zda je seskupení nebo segmentace povolena
- Generování nového identifikátoru zprávy a identifikátoru korelace
- Pořadí, ve kterém jsou zprávy a segmenty vloženy do fronty
- Zda se mají interpretovat názvy lokálních front

Pokud ponecháte pole *Options* nastaveno na výchozí hodnotu (MQPMO_NONE), bude vámi zadaná zpráva mít k sobě přidruženy výchozí kontextové informace.

Také způsob, jakým volání pracuje se synchronizačním body, je určen platformou. Výchozí hodnota řízení synchronizačního bodu je yes v systému z/OS; pro jiné platformy to není žádné.

Kontext
Tím je uveden název popisovače fronty, ze kterého chcete kopírovat informace o kontextu (je-li to požadováno v poli *Options*).

Úvod do kontextu zpráv naleznete v tématu “kontext zprávy” na stránce 37. Informace o použití struktury MQPMO k řízení informací o kontextu ve zprávě naleznete v tématu “Řízení informací o kontextu” na stránce 222.

ResolvedQName
Obsahuje jméno (po vyřešení jakéhokoli jména alias) fronty, která byla otevřena pro přijetí zprávy. Toto je výstupní pole.

Název ResolvedQMgr
Obsahuje název (po vyřešení všech názvů alias) správce front, který je vlastníkem fronty v produktu *ResolvedQName*. Toto je výstupní pole.

MQPMO může také obsahovat pole požadovaná pro distribuční seznamy (viz “Distribuční seznamy” na stránce 226). Chcete-li tuto funkci použít, použije se verze 2 struktury MQPMO. To zahrnuje následující pole:

RecsPresent
Toto pole obsahuje počet front v seznamu distribuce; tj. počet záznamů vložení záznamů (MQPMR) a odpovídající záznamy odpovědí (MQRR).

Hodnota, kterou zadáte, může být stejná jako počet záznamů objektů poskytnutých v MQOPEN. Je-li však hodnota nižší než počet záznamů objektu poskytnutých v rámci volání MQOPEN, nebo pokud nezadáte žádné záznamy s vložením zprávy, budou hodnoty front, které nejsou definovány, převzaty z výchozích hodnot poskytnutých deskriptorem zprávy. Také je-li hodnota větší než počet poskytnutých záznamů o objektu, budou nadbytečné záznamy vložení zpráv ignorovány.

Doporučuje se provést jednu z následujících možností:

- Chcete-li obdržet sestavu nebo odpověď z každého místa určení, zadejte stejnou hodnotu, jaká se objevuje ve struktuře MQOR, a použijte pole MQPMR obsahující pole *MsgId* . Buď inicializujte tato pole *MsgId* na nuly, nebo uveďte MQPMO_NEW_MSG_ID.

  Po vložení zprávy do fronty jsou v MQPMRs; k dispozici hodnoty *MsgId* , které správce front vytvořil; můžete je použít k identifikaci, který cíl je přidružen ke každé sestavě nebo k odpovědi.

- Pokud nechcete přijímat sestavy nebo odpovědi, vyberte jednu z následujících možností:
  1. Chcete-li identifikovat cíle, které selžou okamžitě, můžete stále chtít zadat stejnou hodnotu do pole *RecsPresent* , jak se objevuje ve struktuře MQOR, a poskytnout MQRRs pro identifikaci těchto cílů. Nezadávejte žádné příkazy MQPMRs.
  2. Pokud nechcete identifikovat místa určení, která se nezdařila, zadejte do pole *RecsPresent* nulu a neposkytujte MQPMRs ani MQRRs.

Poznámka: Pokud používáte MQPUT1, musí být počet ukazatelů odezvy záznamu odezvy a Offset záznamu odezvy nula.

Úplný popis záznamů vložení zpráv (MQPMR) a záznamů odpovědí (MQRR) naleznete v části MQPMR a MQRR.

PutMsgRecFields
Označuje, která pole se nacházejí v každém záznamu vložení zprávy (MQPMR). Seznam těchto polí naleznete v části "Použití struktury MQPMR" na stránce 229.

PutMsgRecOffset* a *PutMsgRecPtr
Ukazatelé (obvykle v C) a posuny (obvykle v jazyce COBOL) se používají k adresování záznamů vložení záznamů zpráv (viz "Použití struktury MQPMR" na stránce 229 pro přehled struktury MQPMR).

Pole *PutMsgRecPtr* použijte k uvedení ukazatele na první záznam zprávy Put, nebo pole *PutMsgRecOffset* pro uvedení posunutí prvního záznamu zprávy o vložení. Toto je posun od začátku MQPMO. V závislosti na poli *PutMsgRecFields* zadejte hodnotu, která není null pro *PutMsgRecOffset* nebo *PutMsgRecPtr*.

Posunutí ResponseReca Ptr ResponseRec
Také použijete ukazatele a posuny k adresování záznamů odpovědí (viz "Použití struktury MQRR" na stránce 228 , kde získáte další informace o záznamech odpovědí).

Do pole *ResponseRecPtr* zadejte ukazatel na první záznam odpovědi nebo pole *ResponseRecOffset* , abyste zadali posun prvního záznamu odezvy. Toto je posun od začátku struktury MQPMO. Zadejte nenull hodnotu buď pro *ResponseRecOffset* , nebo *ResponseRecPtr*.

Poznámka: Pokud používáte MQPUT1 k vložení zpráv do rozdělovníku, *ResponseRecPtr* musí být null nebo nula a *ResponseRecOffset* musí být nula.

Verze 3 struktury MQPMO dále obsahuje následující pole:

OriginalMsgManipulátor
Použití tohoto pole může být závislé na hodnotě pole *Akce* . Pokud vkládáte novou zprávu s přidruženými vlastnostmi zprávy, nastavte toto pole na popisovač zprávy, který jste již dříve vytvořili a nastavte vlastnosti. Pokud předáváte, odpovídáte nebo generujete sestavu jako odpověď na dříve načtenou zprávu, bude toto pole obsahovat odkaz na zprávu dané zprávy.

Popisovač NewMsg
Pokud uvedete *NewMsgHandle*, jakékoli vlastnosti vztahující se ke zpracování vlastností přepisu přidružené k *OriginalMsgHandle*. Další informace viz Akce (MQLONG).

Akce
Prostřednictvím tohoto pole můžete určit typ prováděné operace. Možné hodnoty a jejich významy jsou následující:

NOVÁ HODNOTA MQACTP_NEW
Toto je nová zpráva, která nesouvisí s žádným jiným.

MQACTP_FORWARD
Tato zpráva byla načtena dříve a nyní je předávána.

MQACTP_REPLY
Tato zpráva je odpovědí na dříve načtenou zprávu.

SESTAVA MQACTP_REPORT
Tato zpráva je sestava generovaná jako výsledek dříve načtené zprávy.

Další informace viz Akce (MQLONG).

PubLevel
Je-li tato zpráva publikami, můžete toto pole nastavit, abyste určili, které odběry se mají přijmout. Tato publikace bude přijímat pouze odběry s *SubLevel* , které jsou nižší než nebo rovny této hodnotě. Výchozí hodnota je 9, což je nejvyšší úroveň, což znamená, že odběry s libovolným *SubLevel* mohou tuto publikaci přijmout.

Data ve zprávě

Zadejte adresu vyrovnávací paměti, která obsahuje data, v parametru *Buffer* volání MQPUT. Do dat můžete zahrnout cokoliv, co obsahuje data. Množství dat ve zprávách však ovlivňuje výkon aplikace, která je zpracovává.

Maximální velikost dat je určena následujícím způsobem:

- Atribut *MaxMsgLength* správce front
- Atribut *MaxMsgLength* fronty, na kterou stavíte zprávu
- Velikost libovolného záhlaví zprávy přidaného produktem WebSphere MQ (včetně záhlaví nedoručených zpráv, MQDLH a záhlaví distribučního seznamu, MQDH)

Atribut *MaxMsgLength* správce front uchovává velikost zprávy, kterou může správce front zpracovat. Tato hodnota má výchozí hodnotu 100 MB pro všechny produkty WebSphere MQ v V6 nebo vyšší.

Chcete-li určit hodnotu tohoto atributu, použijte volání MQINQ u objektu správce front. Pro velké zprávy můžete tuto hodnotu změnit.

Atribut *MaxMsgLength* fronty určuje maximální velikost zprávy, kterou lze vložit do fronty. Pokusíte-li se vložit zprávu o větší velikosti, než je hodnota tohoto atributu, volání MQPUT se nezdaří. Pokud vkládáte zprávu do vzdálené fronty, maximální velikost zprávy, kterou lze úspěšně uložit, je určena atributem *MaxMsgLength* vzdálené fronty, libovolnými intermediačními přijímajícími frontami, které je zpráva umístěna na trase do místa určení, a použitých kanálů.

Pro operaci MQPUT musí být velikost zprávy menší nebo rovna atributu *MaxMsgLength* jak fronty, tak i správce front. Hodnoty těchto atributů jsou nezávislé, ale doporučuje se nastavit *MaxMsgLength* fronty na hodnotu menší nebo rovnou hodnotě, která má správce front.

Produkt WebSphere MQ přidává informace o záhlaví do zpráv za následujících okolností:

- Když vložíte zprávu do vzdálené fronty, produkt WebSphere MQ přidá do zprávy strukturu záhlaví přenosu (MQXQH). Tato struktura zahrnuje název cílové fronty a jejího vlastníka správce front.
- Nemůže-li produkt WebSphere MQ doručit zprávu do vzdálené fronty, pokusí se zprávu vložit do fronty nedoručených zpráv (undelivered-message). Přidává strukturu MQDLH do zprávy. Tato struktura zahrnuje název fronty místa určení a důvod, proč byla zpráva vložena do fronty nedoručených zpráv.
- Chcete-li odeslat zprávu do více cílových front, produkt WebSphere MQ přidá záhlaví MQDH do zprávy. Popisuje data, která jsou přítomna ve zprávě, která patří do distribučního seznamu, do přenosové fronty. Zvažte tuto volbu při výběru optimální hodnoty pro maximální délku zprávy.
- Je-li zpráva segment nebo zpráva ve skupině, může produkt WebSphere MQ přidat MQMDE.

Tyto struktury jsou popsány v MQDH a MQMDE.

Pokud vaše zprávy mají maximální povolenou velikost pro tyto fronty, přidání těchto záhlaví znamená, že operace put se nezdaří, protože zprávy jsou nyní příliš velké. Chcete-li snížit pravděpodobnost, že operace put selže, postupujte takto:

- Nastavte velikost zpráv menších než atribut *MaxMsgLength* pro fronty přenosu a fronty nedoručených zpráv. Povolte alespoň hodnotu konstanty MQ_MSG_HEADER_LENGTH (více pro velké distribuční seznamy).
- Ujistěte se, že atribut *MaxMsgLength* fronty nedoručených zpráv je nastaven na stejnou hodnotu jako *MaxMsgLength* správce front, který vlastní frontu nedoručených zpráv.

Atributy pro správce front a konstanty fronty zpráv jsou popsány v tématu Atributy pro správce front.

Vložení zpráv: Použití obslužných rutin zpráv

Ve struktuře MQPMO jsou k dispozici dva popisovače zpráv, *OriginalMsgHandle* a *NewMsgHandle*. Vztah mezi těmito manipulátory zpráv je definován hodnotou pole MQPMO *Akce* .

Podrobné informace naleznete v tématu Akce (MQLONG). Ovladač zprávy nemusí být nutně vyžadován, aby bylo možné zprávu vložit. Jeho účelem je přidružit vlastnosti ke zprávě, takže je zapotřebí pouze v případě, že používáte vlastnosti zprávy.

Vložení zpráv do vzdálené fronty

Chcete-li vložit zprávu do vzdálené fronty (tj. do fronty vlastněné jiným správcem fronty, než je ten, ke kterému je aplikace připojena) místo lokální fronty, je jedinou přebytečnou otázkou způsob, jak zadat název fronty při otevření.Tento popis je popsán v tématu “Otevírání vzdálených front” na stránce 215. Není zde žádná změna způsobu použití volání MQPUT nebo MQPUT1 pro lokální frontu.

Další informace o používání vzdálených a přenosových front naleznete v příručce WebSphere MQ distributed-messaging techniques .

Nastavení vlastností zprávy

Volejte funkci MQSETMP pro každou vlastnost, kterou chcete nastavit. Když vložíte zprávu do pole popisovač zprávy a pole akce struktury MQPMO, nastavte ji.

Chcete-li přidružit vlastnosti ke zprávě, musí mít zpráva popisovač zprávy. Vytvořte popisovač zprávy pomocí volání funkce MQCRTMH. Volejte funkci MQSETMP, která určuje tento popisovač zprávy pro každou vlastnost, kterou chcete nastavit. K dispozici je ukázkový program amqsstma.cpro ilustraci použití příkazu MQSETMP.

Pokud se jedná o novou zprávu, když ji vložíte do fronty pomocí příkazu MQPUT nebo MQPUT1, nastavte pole OriginalMsgv MQPMO na hodnotu tohoto manipulátoru zprávy a nastavte pole akce MQPMO na hodnotu MQACTP_NEW (jedná se o výchozí hodnotu).

Pokud se jedná o zprávu, kterou jste již dříve získali, a vy nyní předáváte nebo odpovídáte na ni nebo na odeslání sestavy v odpovědi, umístěte původní popisovač zprávy do pole OriginalMsgobslužné rutiny MQPMO a nový popisovač zprávy v poli Popisovač NewMsg. Podle potřeby nastavte pole Akce na hodnotu MQACTP_FORWARD, MQACTP_REPLY nebo MQACTP_REPORT.

Máte-li vlastnosti v záhlaví MQRFH2 ze zprávy, kterou jste již dříve získali, můžete je převést na vlastnosti obslužné rutiny zpráv pomocí volání MQBUFMH.

Zakládáte-li zprávu do fronty ve správci front na úrovni dřívější než WebSphere MQ verze 7.0, která nemůže zpracovat vlastnosti zprávy, můžete nastavit parametr PropertyControl v definici kanálu a určit, jak mají být vlastnosti zpracovávány.

Řízení informací o kontextu

Použijete-li volání MQPUT nebo MQPUT1 k vložení zprávy do fronty, můžete určit, že správce front má přidat některé výchozí kontextové informace do deskriptoru zpráv. Aplikace, které mají odpovídající úroveň oprávnění, mohou přidávat další informace o kontextu. Pole voleb ve struktuře MQPMO můžete použít k řízení informací o kontextu.

Chcete-li řídit informace o kontextu, použijte pole *Options* ve struktuře MQPMO.

Pokud tomu tak není, správce front přepíše informace o kontextu, které mohou již být v deskriptoru zpráv, s informacemi o identitě a kontextu, které vygenerovala pro vaši zprávu. To je stejné jako uvedení volby MQPMO_DEFAULT_CONTEXT.Tyto výchozí informace o kontextu můžete chtít při vytváření nové zprávy (například při zpracování vstupu uživatele z dotazové obrazovky).

Pokud nechcete k dané zprávě přidružit žádné kontextové informace, použijte volbu MQPMO_NO_CONTEXT. Při vkládání zprávy bez kontextu jsou jakékoli kontroly oprávnění provedené produktem IBM WebSphere MQ prováděny s použitím prázdného ID uživatele. Prázdné ID uživatele nemůže být přiřazeno explicitnímu oprávnění k prostředkům produktu IBM WebSphere MQ , ale je

zpracováno jako člen speciální skupiny 'nobody'. Další podrobnosti o speciální skupině nobodynaleznete v tématu Referenční informace o rozhraní služeb instalovatelných služeb.

Pokud nechcete k dané zprávě přidružit žádné kontextové informace, použijte volbu MQPMO_NO_CONTEXT.

Následující části tohoto tématu popisují použití kontextu identity, kontextu uživatele a všech kontextů.

Předání kontextu identity

Obecně řečeno, programy by měly předávat informace o kontextu identity ze zprávy do zprávy okolo aplikace, dokud data nedosáhne svého konečného cíle.

Programy by měly při každé změně dat změnit informace o kontextu původu. Aplikace, které chtějí změnit nebo nastavit jakékoli informace o kontextu, však musí mít odpovídající úroveň oprávnění. Správce front zkontroluje toto oprávnění, pokud aplikace otevřou fronty. Musí mít oprávnění k použití příslušných kontextových voleb pro volání MQOPEN.

Pokud vaše aplikace obdrží zprávu, zpracuje data z této zprávy, pak vloží změněná data do jiné zprávy (případně pro zpracování jinou aplikací), aplikace musí předat informace o kontextu identity z původní zprávy do nové zprávy. Můžete povolit správci front vytvořit informace o kontextu původu.

Chcete-li uložit informace o kontextu z původní zprávy, použijte volbu MQOO_SAVE_ALL_CONTEXT, když otevřete frontu pro získání zprávy. To je dodatkem k dalším volbám, které používáte při volání MQOPEN. Všimněte si však, že informace o kontextu nelze uložit, pokud pouze procházíte zprávou.

Při vytvoření druhé zprávy:

- Otevřete frontu s použitím volby MQOO_PASS_IDENTITY_CONTEXT (navíc k volbě MQOO_OUTPUT).
- V poli *Context* struktury voleb vložení zprávy zadejte popisovač fronty, ze které jste uložili informace o kontextu.
- V poli *Options* struktury příkazu put-message určete volbu MQPMO_PASS_IDENTITY_CONTEXT.

Předání kontextu uživatele

Nemůžete se rozhodnout předat pouze kontext uživatele. Chcete-li předat kontext uživatele při vložení zprávy, uveďte MQPMO_PASS_ALL_CONTEXT. Všechny vlastnosti v kontextu uživatele jsou předávány stejným způsobem jako kontext původu.

Je-li provedeno volání MQPUT nebo MQPUT1 a kontext je předáván, všechny vlastnosti v kontextu uživatele jsou předány z načtené zprávy do zprávy vložení. Všechny vlastnosti kontextu uživatele, které vkládá aplikace do aplikace, se umístí s jejich původními hodnotami. Všechny vlastnosti kontextu uživatele, které aplikace uvedení do provozu odstranily, jsou obnoveny ve zprávě vložení. Zadrží se všechny vlastnosti kontextu uživatele, které vkládá aplikace vkládání do zprávy.

Předávání všech kontextů

Pokud vaše aplikace obdrží zprávu a vloží do jiné zprávy data zprávy (nezměněná), aplikace musí předat všechny informace o kontextu (identita, původ a uživatel) z původní zprávy do nové zprávy. Příkladem aplikace, která by mohla být tato, je modul pro přesouvání zpráv, který přesouvá zprávy z jedné fronty do jiné.

Postupujte stejným postupem jako při předávání kontextu identity, kromě toho, že jste použili volbu MQOTE_PASS_ALL_CONTEXT a volbu vložení zprávy MQPMO_PASS_ALL_CONTEXT.

Nastavení kontextu identity

Chcete-li nastavit informace o kontextu identity pro zprávu:

- Otevřete frontu s použitím volby MQOO_SET_IDENTITY_CONTEXT.
- Vložte zprávu do fronty a zadejte volbu MQPMO_SET_IDENTITY_CONTEXT. V deskriptoru zpráv zadejte jakékoli informace o kontextu identity, které požadujete.

Poznámka: Nastavíte-li některé (ale ne všechny) pole kontextu identity pomocí voleb MQOO_SET_IDENTITY_CONTEXT a MQPMO_SET_IDENTITY_CONTEXT, je důležité si uvědomit, že správce front nenastavuje žádná jiná pole.

Chcete-li upravit kteroukoli z voleb kontextu zprávy, musíte mít příslušná oprávnění k vydávání volání. Chcete-li například použít funkci MQOO_SET_IDENTITY_CONTEXT nebo MQPMO_SET_IDENTITY_CONTEXT, musíte mít oprávnění `+setid` .

Nastavení kontextu uživatele

Chcete-li nastavit vlastnost v kontextu uživatele, nastavte pole Kontext deskriptoru vlastnosti zprávy (MQPD) na hodnotu MQPD_USER_CONTEXT při volání funkce MQSETMP.

K nastavení vlastnosti v kontextu uživatele nepotřebujete žádné speciální oprávnění. Kontext uživatele nemá žádné volby kontextu MQOO_SET_ * nebo MQPMO_SET_ *.

Nastavení celého kontextu

Chcete-li nastavit jak identitu, tak informace o kontextu původu pro zprávu, postupujte takto:

1. Otevřete frontu s použitím volby MQOO_SET_ALL_CONTEXT.
2. Vložte zprávu do fronty uvedením volby MQPMO_SET_ALL_CONTEXT. V deskriptoru zpráv zadejte libovolnou informaci o identitě a původu, kterou požadujete.

Pro každý typ nastavení kontextu je zapotřebí příslušné oprávnění.

Související pojmy

“kontext zprávy” na stránce 37
Informace *Kontext zprávy* umožňují aplikaci, která načte zprávu, zjistit informace o odesílateli zprávy.

Související odkazy

“Volby MQOPEN vztahující se ke kontextu zprávy” na stránce 214
Chcete-li mít možnost přidružit kontextové informace ke zprávě, když ji vložíte do fronty, musíte při otevření fronty použít jednu z voleb kontextu zprávy.

Vložení jedné zprávy do fronty pomocí volání MQPUT1

Použijte volání MQPUT1 , chcete-li zavřít frontu okamžitě poté, co jste do ní vložili jednu zprávu. Například serverová aplikace pravděpodobně používá volání MQPUT1 , když odesílá odpověď na každou z různých front.

Hodnota MQPUT1 je funkčně ekvivalentní volání MQOPEN následované operací MQPUT následovaným příkazem MQCLOSE. Jediným rozdílem v syntaxi pro volání MQPUT a MQPUT1 je to, že pro příkaz MQPUT uvedete popisovač objektu, zatímco pro MQPUT1 určujete strukturu deskriptoru objektu (MQOD), jak je definováno v MQOPEN (viz “Označení objektů (struktura MQOD)” na stránce 208). Důvodem je skutečnost, že je třeba předat informace o volání MQPUT1 ke frontě, kterou má otevřít, zatímco při volání funkce MQPUT se musí fronta již otevřít.

Jako vstup do volání MQPUT1 je třeba dodat:

- Popisovač připojení.

- Popis objektu, který chcete otevřít. Tato hodnota je ve formě struktury deskriptoru objektu (MQOD).
- Popis zprávy, kterou chcete vložit do fronty. To je ve formě struktury deskriptoru zpráv (MQMD).
- Řídicí informace ve formě struktury voleb put-message (MQPMO).
- Délka dat obsažených ve zprávě (MQLONG).
- Adresa dat zprávy.

Výstup příkazu MQPUT1 je následující:

- Kód dokončení
- Kód příčiny

Pokud je volání úspěšně dokončeno, vrací také strukturu vaší volby a strukturu deskriptoru zprávy. Volání upravuje strukturu voleb tak, aby zobrazovala název fronty a správce front, do kterého byla zpráva odeslána. Pokud požadujete, aby správce front generoval jedinečnou hodnotu pro identifikátor zprávy, kterou umisťujete (zadáním binární nuly v poli *MsgId* struktury MQMD), zavolání vloží hodnotu do pole *MsgId* před vrácením této struktury zpět.

Poznámka: MQPUT1 nelze použít s názvem modelové fronty, avšak po otevření modelové fronty můžete k dynamické frontě vydat příkaz MQPUT1 .

Je šest vstupních parametrů pro MQPUT1 :

Hconn
Jedná se o manipulátor připojení. Pro aplikace CICS můžete zadat konstantu MQHC_DEF_HCONN (která má hodnotu nula), nebo použít manipulátor připojení vrácený voláním MQCONN nebo MQCONNX. Pro ostatní programy vždy použijte obslužnou rutinu připojení vrácenou voláním MQCONN nebo MQCONNX.

ObjDesc
Jedná se o strukturu deskriptoru objektu (MQOD).

Do polí *ObjectName* a *ObjectQMgrName* zadejte název fronty, do které chcete vložit zprávu, a název správce front, který je vlastníkem této fronty.

Pole *DynamicQName* je pro volání MQPUT1 ignorováno, protože nemůže používat modelové fronty.

Použijte pole *AlternateUserId* , chcete-li navrhnout alternativní identifikátor uživatele, který má být použit pro testovací oprávnění k otevření fronty.

MsgDesc
Jedná se o strukturu deskriptoru zpráv (MQMD). Stejně jako v případě volání MQPUT použijte tuto strukturu k definování zprávy, kterou vkládáte do fronty.

PutMsgOpts
Jedná se o strukturu voleb vložení zprávy (MQPMO). Použijte jej, jak byste měli pro volání MQPUT (viz “Určení voleb pomocí struktury MQPMO” na stránce 218).

Je-li pole *Options* nastaveno na hodnotu nula, správce front používá vlastní ID uživatele při provádění testů pro oprávnění pro přístup k frontě. Správce front bude také ignorovat alternativní identifikátor uživatele zadaný v poli *AlternateUserId* struktury MQOD.

BufferLength
Toto je délka vaší zprávy.

Buffer
Jedná se o oblast vyrovnávací paměti, která obsahuje text vaší zprávy.

Pokud používáte klastry, funkce MQPUT1 bude fungovat, jako by bylo v platnosti vlastnost MQOO_BIND_NOT_FIXED. Aplikace musí používat vyřešená pole ve struktuře MQPMO místo struktury MQOD k určení, kam byla zpráva odeslána. Další informace naleznete v tématu Konfigurace klastru správce front .

Je uveden popis volání MQPUT1 v umístění MQPUT1.

Distribuční seznamy

Není podporováno na produktu WebSphere MQ pro z/OS. Distribuční seznamy umožňují vložit zprávu do více míst určení v rámci jednoho volání MQPUT nebo MQPUT1 .Jedno volání MQOPEN může otevřít více front a jediné volání MQPUT může poté vložit zprávu do každé z těchto front. Některé generické informace ze struktur MQI použitých pro tento proces mohou být nahrazeny specifickými informacemi týkajícími se jednotlivých míst určení zahrnutých do rozdělovníku.

V 7.5.0.8

Upozornění: Distribuční seznamy nepodporují použití alias fronta, které odkazují na objekty tématu. Od Version 7.5.0, Fix Pack 8, jestliže alias fronta odkazuje na objekt tématu v distribučním seznamu, IBM WebSphere MQ vrací MQRC_ALIAS_BASE_Q_TYPE_ERROR.

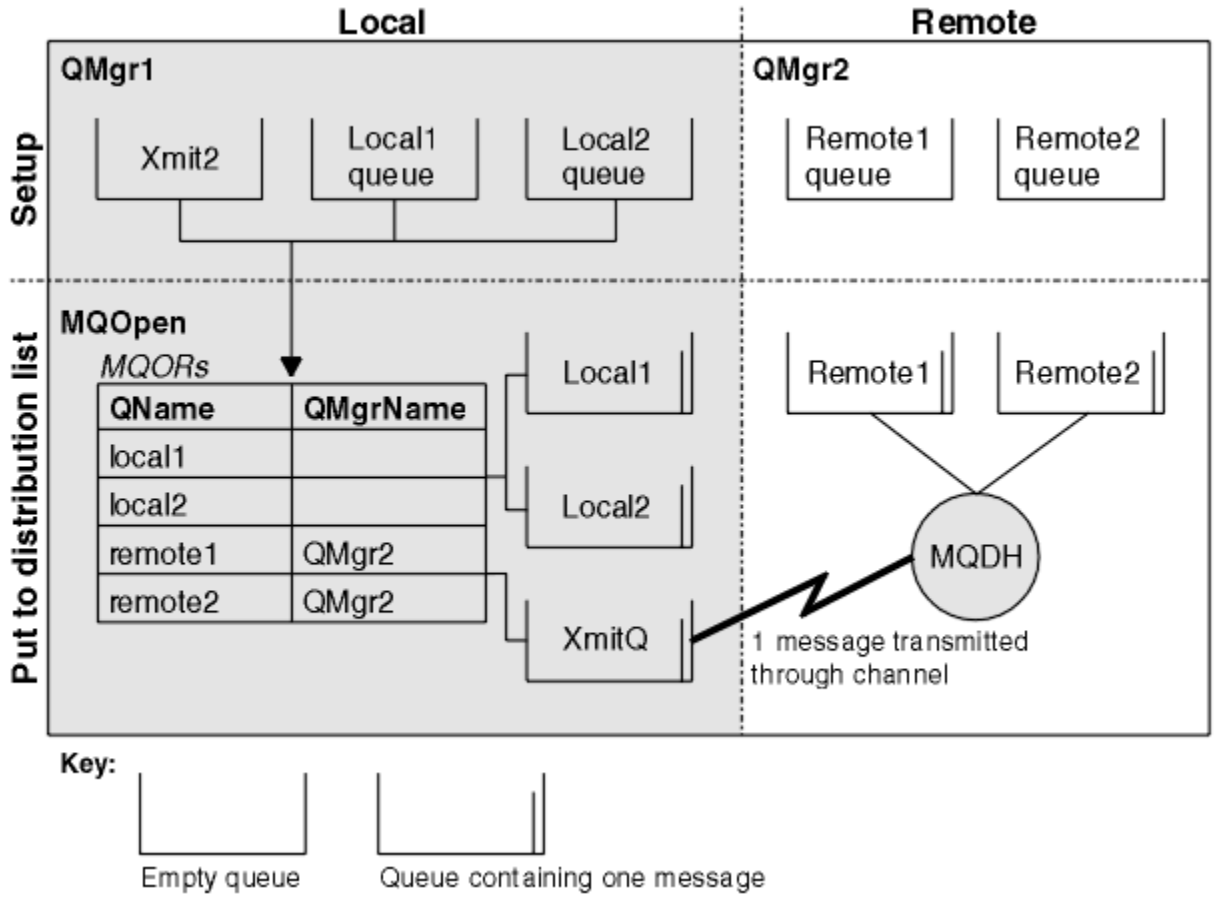
Je-li vydáno volání MQOPEN, jsou obecné informace převzaty z deskriptoru objektu (MQOD). Pokud uvedete MQOD_VERSION_2 do pole *Version* a hodnotu větší než nula v poli *RecsPresent* , *Hobj* může být definováno jako popisovač seznamu (jedné nebo více front), spíše než fronty. V tomto případě jsou specifické informace poskytnuty prostřednictvím záznamů objektů (MQORs), které uvádějí podrobnosti o místě určení (to znamená *ObjectName* a *ObjectQMgrName*).

Manipulátor objektu (*Hobj*) je předán volání MQPUT, což umožňuje vložení do jedné fronty a nikoli do jediné fronty.

Je-li zpráva vložena do fronty (MQPUT), jsou generické informace převzaty ze struktury volby vložení zprávy (MQPMO) a MQMD (Message Descriptor). Specifické informace jsou uvedeny ve formě záznamů vložení zpráv (MQPMRs).

Záznamy odpovědí (MQRR) mohou přijmout kód dokončení a kód příčiny specifický pro každou cílovou frontu.

Příkaz Obrázek 29 na stránce 226 ukazuje, jak distribuční seznamy fungují.



Obrázek 29. Jak distribuční seznamy fungují

Otevírání distribučních seznamů

Použijte volání MQOPEN k otevření rozdělovníku a použijte volby volání k uvedení toho, co chcete dělat se seznamem.

Jako vstup pro volání MQOPEN je třeba zadat:

- Popisovač připojení (viz “Vložení zpráv do fronty” na stránce 216 pro popis)
- Generické informace ve struktuře deskriptoru objektu (MQOD)
- Název každé fronty, kterou chcete otevřít, pomocí struktury záznamů objektů (MQOR)

Výstup z MQOPEN je:

- Popisovač objektu, který představuje váš přístup k rozdělovníku
- Obecný kód dokončení
- Generický kód příčiny
- Záznamy odpovědí (volitelné), obsahující kód dokončení a důvod pro každé místo určení

Použití struktury MQOD

Strukturu MQOD použijte k identifikaci front, které chcete otevřít.

Chcete-li definovat distribuční seznam, musíte zadat MQOD_VERSION_2 do pole *Version* , hodnota větší než nula v poli *RecsPresent* a MQOT_Q v poli *ObjectType* . Popis všech polí struktury MQOD najdete v tématu MQOD .

Použití struktury MQOR

Zadejte strukturu MQOR pro každý cíl.

Struktura obsahuje názvy cílové fronty a správce front. Pole *ObjectName* a *ObjectQMgrName* v MQOD se nepoužívají pro distribuční seznamy. Musí existovat jeden nebo více záznamů objektů. Je-li pole *ObjectQMgrName* ponecháno prázdné, použije se lokální správce front. Další informace o těchto polích viz ObjectName a ObjectQMgrName .

Cílové fronty lze určit dvěma způsoby:

- Použitím pole offsetu *ObjectRecOffset*.

  V takovém případě aplikace musí deklarovat vlastní strukturu obsahující strukturu MQOD, za kterou bude následovat pole záznamů MQOR (s tolika prvky pole, kolik je potřeba), a nastavit proměnnou *ObjectRecOffset* na posun prvního prvku v poli od začátku MQOD. Ujistěte se, že je tento posun správný.

  Doporučuje se použití vestavěných systémových prostředků poskytnutých programovacím jazykem, pokud jsou k dispozici ve všech prostředích, ve kterých je aplikace spuštěna. Následující kód ilustruje tuto techniku pro programovací jazyk COBOL:

```
01  MY-OPEN-DATA.
    02 MY-MQOD.
       COPY CMQODV.
    02 MY-MQOR-TABLE OCCURS 100 TIMES.
       COPY CMQORV.
    MOVE LENGTH OF MY-MQOD TO MQOD-OBJECTRECOFFSET.
```

  Případně můžete použít konstantní MQOD_CURRENT_LENGTH, pokud programovací jazyk nepodporuje nezbytná vestavěná zařízení ve všech dotčených prostředích. Následující kód ilustruje tuto techniku:

```
01  MY-MQ-CONSTANTS.
    COPY CMQV.
01  MY-OPEN-DATA.
    02 MY-MQOD.
       COPY CMQODV.
    02 MY-MQOR-TABLE OCCURS 100 TIMES.
```

```
         COPY CMQORV.
      MOVE MQOD-CURRENT-LENGTH TO MQOD-OBJECTRECOFFSET.
```

To však funguje správně pouze v případě, že struktura MQOD a pole záznamů MQOR jsou souvislé; pokud kompilátor vkládá mezi MQOD a MQOR pole přeskočených bajtů, je nutné je přidat k hodnotě uložené v produktu *ObjectRecOffset*.

Použití *ObjectRecOffset* se doporučuje pro programovací jazyky, které nepodporují datový typ ukazatele, nebo které implementují datový typ ukazatele takovým způsobem, který není přenosný do různých prostředí (například programovací jazyk COBOL).

- Použitím pole ukazatele *ObjectRecPtr*.

  V takovém případě může aplikace deklarovat pole struktury MQOR odděleně od struktury MQOD a nastavit *ObjectRecPtr* na adresu pole. Následující kód ilustruje tuto techniku pro programovací jazyk C:

```
MQOD MyMqod;
MQOR MyMqor[100];
MyMqod.ObjectRecPtr = MyMqor;
```

  Použití *ObjectRecPtr* se doporučuje pro programovací jazyky, které podporují datový typ ukazatele způsobem, který je přenosný do různých prostředí (například programovací jazyk C).

Zvolená technika musí používat jeden z produktů *ObjectRecOffset* a *ObjectRecPtr*; volání selže s kódem příčiny MQRC_OBJECT_RECORS_ERROR, pokud jsou obě nulové nebo obě jsou nenulové.

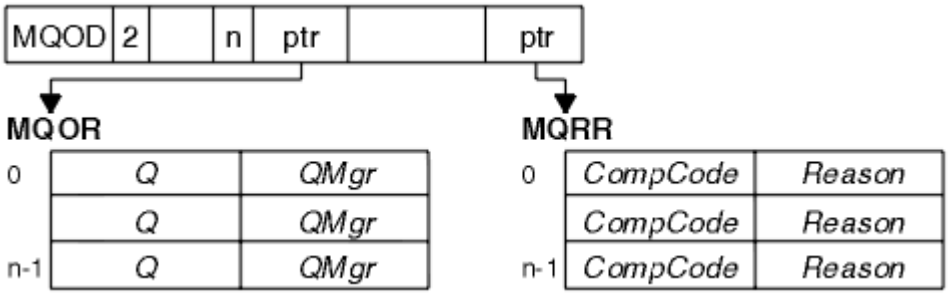
Použití struktury MQRR

Tyto struktury jsou specifické pro cíl; každý záznam odpovědi obsahuje pole *CompCode* a *Reason* pro každou frontu distribučního seznamu. Tuto strukturu musíte použít k tomu, abyste mohli rozlišovat, kde se vyskytují nějaké problémy.

Pokud například obdržíte kód příčiny MQRC_MULTIPLE_REASONS a váš distribuční seznam obsahuje pět cílových front, nebudete vědět, do kterých front se tyto problémy vztahují, pokud tuto strukturu nepoužijete. Pokud však máte kód dokončení a kód příčiny pro každé místo určení, můžete chyby snadněji lokalizovat.

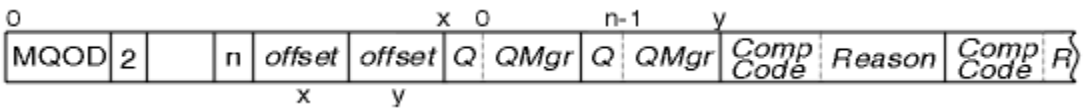
Další informace o struktuře MQRR najdete v tématu MQRR .

Obrázek 30 na stránce 228 ukazuje, jak můžete otevřít distribuční seznam v C.



Obrázek 30. Otevření distribučního seznamu v jazyce C

Produkt Obrázek 31 na stránce 228 zobrazuje, jak můžete otevřít distribuční seznam v jazyce COBOL.



Obrázek 31. Otevření distribučního seznamu v jazyce COBOL

Použití voleb MQOPEN

Při otevírání distribučního seznamu můžete určit následující volby:

- MQOOK_VÝSTUP
- MQOO_FAIL_IF_QUIESCING (volitelné)
- MQONE_ALTERNATE_USER_AUTHORITY (volitelné)
- MQOO_ * _CONTEXT (volitelné)

Popis těchto voleb najdete v části “Otevírání a zavírání objektů” na stránce 206 .

Vložení zpráv do distribučního seznamu

Chcete-li vložit zprávy do distribučního seznamu, můžete použít příkaz MQPUT nebo MQPUT1.

Jako vstup musíte dodat:

- Popisovač připojení (viz “Vložení zpráv do fronty” na stránce 216 pro popis).
- Popisovač objektu. Je-li distribuční seznam otevřen pomocí funkce MQOPEN, produkt *Hobj* umožňuje pouze vložení do seznamu.
- Struktura deskriptoru zpráv (MQMD). Popis této struktury viz MQMD .
- Řídicí informace v podobě struktury volby put-message (MQPMO). Informace o vyplnění polí struktury MQPMO naleznete v části “Určení voleb pomocí struktury MQPMO” na stránce 218 .
- Řídicí informace ve formátu záznamů vložení zpráv (MQPMR).
- Délka dat obsažených ve zprávě (MQLONG).
- Samotná data zprávy.

Výstup je:

- Kód dokončení
- Kód příčiny
- Záznamy odpovědí (volitelné)

Použití struktury MQPMR

Tato struktura je volitelná a uvádí informace specifické pro místo určení pro některá pole, která byste mohli chtít identifikovat jinak než hodnoty, které jsou již identifikovány v MQMD.

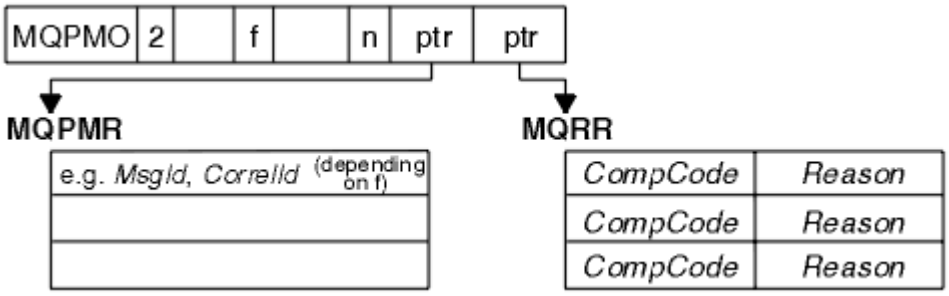
Popis těchto polí naleznete v tématu MQPMR.

Obsah jednotlivých záznamů závisí na informacích uvedených v poli *PutMsgRecFields* MQPMO. Příklad: v ukázkovém programu AMQSPTL0.C (viz “Ukázkový program Distribuční seznam” na stránce 122 pro popis) zobrazující použití distribučních seznamů, ukázka zvolí zadání hodnot pro *MsgId* a *CorrelId* v MQPMR. Tato část ukázkového programu vypadá takto:

```
typedef struct
{
MQBYTE24 MsgId;
MQBYTE24 CorrelId;
} PutMsgRec;
...
/**********************
MQLONG PutMsgRecFields=MQPMRF_MSG_ID | MQPMRF_CORREL_ID;
```

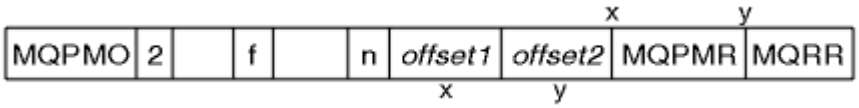
To znamená, že *MsgId* a *CorrelId* jsou poskytnuty pro každé místo určení rozdělovníku. Záznamy Put Message Records jsou poskytovány jako pole.

Obrázek 32 na stránce 230 ukazuje, jak můžete vložit zprávu do rozdělovníku v C.



Obrázek 32. Vložení zprávy do distribučního seznamu v jazyce C

Produkt Obrázek 33 na stránce 230 zobrazuje, jak můžete vložit zprávu do distribučního seznamu v jazyce COBOL.



Obrázek 33. Vložení zprávy do distribučního seznamu v jazyce COBOL

Použití příkazu MQPUT1

Používáte-li MQPUT1, zvažte následující body:

1. Hodnoty polí *ResponseRecOffset* a *ResponseRecPtr* musí být null nebo nula.
2. Záznamy odpovědí, je-li to nutné, musí být adresovány z MQOD.

Některé případy, kdy volání vložení selžou

Změní-li se některé atributy fronty použitím příkazu FORCE u příkazu během intervalu mezi zadáním volání MQOPEN a voláním MQPUT, volání MQPUT selže a vrátí kód příčiny MQRC_OBJECT_CHANGED.

Správce front označí obslužnou rutinu objektu jako neplatnou. K tomu dojde také v případě, že dojde ke zpracování změn během zpracování volání MQPUT1 nebo v případě, že se změny použijí pro každou frontu, na kterou je název fronty vyřešen. Atributy, které ovlivňují popisovač tímto způsobem, jsou uvedeny v popisu volání MQOPEN v MQOPEN. Pokud vaše volání vrátí kód příčiny MQRC_OBJECT_CHANGED, zavřete frontu a znovu ji otevřete a poté zkuste zprávu vložit znovu.

Jsou-li operace vložení zakázány pro frontu, na kterou se pokoušíte vložit zprávy (nebo frontu, na kterou se název fronty rozlišuje), volání MQPUT nebo MQPUT1 selže a vrátí kód příčiny MQRC_PUT_INHIBITED. Pokud se později pokusíte o telefonát, může být zpráva úspěšně vložena, je-li návrh aplikace takový, že jiné programy pravidelně mění atributy front.

Je-li fronta, do které se pokoušíte vložit zprávu, zaplněna, volání MQPUT nebo MQPUT1 selže a vrátí MQRC_Q_FULL.

Byla-li odstraněna dynamická fronta (dočasná nebo trvalá), volání MQPUT s použitím dříve získaného manipulátoru objektu selže a vrátí kód příčiny MQRC_Q_DELETED. V této situaci je dobrým zvykem zavřít popisovač objektu, protože již pro vás není žádný jiný způsob použití.

V případě distribučních seznamů se může v jednom požadavku vyskytnout více kódů dokončení a kódů příčiny. Nelze je zpracovat pouze pomocí výstupních polí *CompCode* a *Reason* v MQOPEN a MQPUT.

Použijete-li distribuční seznamy k umístění zpráv do více míst určení, obsahují záznamy odpovědí specifické *CompCode* a *Reason* pro každý cíl. Obdržíte-li kód dokončení MQCC_FAILED, žádná zpráva nebyla úspěšně vložena do žádné cílové fronty. Je-li kód dokončení MQCC_WARNING, zpráva se úspěšně umístí do jedné nebo více cílových front. Pokud obdržíte návratový kód MQRC_MULTIPLE_REASONS, nejsou všechny kódy příčiny všechny stejné pro všechny cíle. Proto je doporučeno použít strukturu MQRR tak, abyste mohli určit, které fronty nebo fronty způsobily chybu a důvody pro každou z nich.

Získávání zpráv z fronty

Tyto informace použijte k získání informací o získávání zpráv z fronty.

Zprávy z fronty lze získat dvěma způsoby:

1. Můžete odebrat zprávu z fronty, aby ji již ostatní programy nevidět.
2. Můžete zkopírovat zprávu a ponechat původní zprávu ve frontě. To se označuje jako *procházení*. Zprávu můžete odebrat, jakmile ji prohlédnou.

V obou případech je třeba použít volání MQGET, ale nejprve musí být aplikace připojena ke správci front a musíte použít volání MQOPEN k otevření fronty (pro vstup, procházení nebo obojí). Tyto operace jsou popsány v "Připojování k správci front a odpojování od něj" na stránce 198 a "Otevírání a zavírání objektů" na stránce 206.

Po otevření fronty můžete opakovaně používat volání MQGET k procházení nebo odebírání zpráv ve stejné frontě. Po dokončení získávání všech zpráv, které chcete z fronty, zavolejte funkci MQCLOSE.

Chcete-li zjistit více o získávání zpráv z fronty, použijte následující odkazy:

- "Získávání zpráv z fronty pomocí volání MQGET" na stránce 231
- "Pořadí, ve kterém jsou zprávy načítány z fronty" na stránce 235
- "Získání konkrétní zprávy" na stránce 246
- "Zlepšení výkonu přechodných zpráv" na stránce 247
- "Zpracování zpráv větších než 4 MB" na stránce 251
- "Čekání na zprávy" na stránce 256
-
- "Vynechání odvolání" na stránce 257
- "Převod dat aplikace" na stránce 259
- "Procházení zpráv ve frontě" na stránce 260
- "Některé případy, kdy se volání MQGET nezdaří" na stránce 265

Související pojmy

"Přehled rozhraní fronty zpráv" na stránce 187
Zde jsou uvedeny informace o komponentách rozhraní MQI (Message Queue Interface).

"Připojování k správci front a odpojování od něj" na stránce 198
Chcete-li používat programovací služby produktu WebSphere MQ , musí mít program připojení ke správci front. V této části se dozvíte, jak se připojit k správci front a odpojit je od správce front.

"Otevírání a zavírání objektů" na stránce 206
Tyto informace poskytují náhled na otevření a zavření objektů produktu WebSphere MQ .

"Vložení zpráv do fronty" na stránce 216
V této části se dozvíte, jak vložit zprávy do fronty.

"Inquaning about a nastavení atributů objektu" na stránce 307
Atributy jsou vlastnosti, které definují charakteristiky objektu produktu WebSphere MQ .

"Potvrzení a zálohování jednotek práce" na stránce 310
Tyto informace popisují, jak potvrdit a zazálohovat všechny zotavitelné operace get a put, které se vyskytly v jednotce práce.

"Spuštění aplikací produktu IBM WebSphere MQ pomocí spouštěčů" na stránce 316
Informace o spouštěčích a o tom, jak spustit aplikace IBM WebSphere MQ pomocí spouštěčů.

"Práce s MQI a klastry" na stránce 332
Existují speciální volby na voláních a návratových kódech, které se vztahují ke klastrování.

Získávání zpráv z fronty pomocí volání MQGET

Volání MQGET získá zprávu z otevřené lokální fronty. Nelze získat zprávu z fronty na jiném systému.

Jako vstup pro volání MQGET je třeba dodat:

- Popisovač připojení.
- Popisovač fronty.
- Popis zprávy, kterou chcete získat z fronty. Tato struktura je ve formě struktury deskriptoru zpráv (MQMD).
- Řízení informací ve formě struktury MQGMO (Get Message Options).
- Velikost vyrovnávací paměti, kterou jste přiřadili k zadržení zprávy (MQLONG).
- Adresa úložiště, do kterého se má vložit zpráva.

Výstup z MQGET je:

- Kód příčiny
- Kód dokončení
- Zpráva ve vámi zadané oblasti vyrovnávací paměti, pokud je volání úspěšně dokončeno.
- Vaše struktura voleb, upravená tak, aby zobrazovala název fronty, ze které byla zpráva načtena.
- Struktura deskriptoru zpráv s obsahem polí upravených tak, aby popisovala zprávu, která byla načtena.
- Délka zprávy (MQLONG)

V MQGETje uveden popis volání MQGET.

Následující oddíly popisují informace, které musíte dodat jako vstup pro volání MQGET.

- “Určení manipulátorů připojení” na stránce 232
- “Popisování zpráv pomocí struktury MQMD a volání MQGET” na stránce 232
- “Určení voleb MQGET s použitím struktury MQGMO” na stránce 233
- “Určení velikosti oblasti vyrovnávací paměti” na stránce 235

Určení manipulátorů připojení

V případě systému CICS v aplikacích 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 použijte obslužnou rutinu připojení vrácenou voláním MQCONN nebo MQCONNX.

Použijte popisovač fronty (*Hobj*), který je vrácen při volání operace MQOPEN.

Popisování zpráv pomocí struktury MQMD a volání MQGET

Chcete-li identifikovat zprávu, kterou chcete získat z fronty, použijte strukturu deskriptoru zpráv (MQMD).

Jedná se o vstupní/výstupní parametr pro volání MQGET. Existuje úvod do vlastností zprávy, které MQMD popisuje v produktu “Zprávy produktu IBM WebSphere MQ” na stránce 9, a v MQMDje popis struktury samotné.

Pokud víte, kterou zprávu chcete získat z fronty, prohlédněte si téma “Získání konkrétní zprávy” na stránce 246.

Pokud do konkrétní zprávy nezadáte žádnou zprávu, příkaz MQGET načte ve frontě zprávu *first* . “Pořadí, ve kterém jsou zprávy načítány z fronty” na stránce 235 popisuje, jak je priorita zprávy, atribut *MsgDeliverySequence* fronty a volba MQGMO_LOGICAL_ORDER určují pořadí zpráv ve frontě.

Poznámka: Chcete-li použít příkaz MQGET více než jednou (například při procházení zpráv ve frontě), je třeba po každém volání nastavit pole *MsgId* a *CorrelId* této struktury na hodnotu null. Tato akce vymaže tato pole identifikátorů zprávy, která byla načtena.

Pokud však chcete seskupit zprávy, *GroupId* musí být stejné pro zprávy ve stejné skupině, takže volání vypadá pro zprávu mající stejné identifikátory jako předchozí zpráva, aby se celá skupina mohla vytvořit.

Určení voleb MQGET s použitím struktury MQGMO

Struktura MQGMO je vstupní/výstupní proměnnou pro předání voleb do volání MQGET. Následující sekce vám pomohou dokončit některá pole této struktury.

K dispozici je popis struktury MQGMO v produktu MQGMO.

StrucId
StrucId je čtyřznakové pole použité k identifikaci struktury jako struktury voleb pro získání zprávy. Vždy zadejte MQGMO_STRUC_ID.

Version
*Version*popisuje číslo verze struktury. MQGMO_VERSION_1 je výchozí hodnota. Chcete-li použít pole verze 2 nebo načítat zprávy v logickém pořadí, zadejte MQGMO_VERSION_2. Chcete-li použít pole verze 3 nebo načítat zprávy v logickém pořadí, zadejte MQGMO_VERSION_3. Funkce MQGMO_CURRENT_VERSION nastaví aplikaci tak, aby používala nejnovější úroveň.

Options
V rámci vašeho kódu můžete vybrat volby v libovolném pořadí; každá volba je v poli *Options* reprezentována trochou.

Ovládací prvky pole *Options* :

- Určuje, zda volání MQGET čeká na příchod zprávy do fronty před jejím dokončením (viz “Čekání na zprávy” na stránce 256).
- Zda je operace získání zahrnuta do pracovní jednotky.
- Určuje, zda je trvalá zpráva načtena mimo synchronizační bod, což umožňuje rychlé zasílání zpráv.
- V produktu WebSphere MQ for z/OS, zda je načtená zpráva označena jako přeskočení odvolání (viz “Vynechání odvolání” na stránce 257).
- Určuje, zda je zpráva odebrána z fronty, nebo pouze procházená
- Zda se má vybrat zpráva pomocí kurzoru pro procházení nebo jiných kritérií výběru
- Zda je volání úspěšné i v případě, že zpráva je delší než vaše vyrovnávací paměť
- V produktu WebSphere MQ for z/OS, zda má být volání povoleno dokončit. Tato volba také nastaví signál informující o tom, že chcete být upozorněni při doručení zprávy
- Zda se volání nezdaří, je-li správce front ve stavu uvedení do klidového stavu
- V produktu WebSphere MQ for z/OSbez ohledu na to, zda se volání nezdaří, je-li připojení ve stavu uvedení do klidového stavu
- Zda se vyžaduje převod dat zpráv aplikace (viz “Převod dat aplikace” na stránce 259)
- Pořadí, ve kterém jsou zprávy a (kromě WebSphere MQ for z/OS) segmenty načteny z fronty.
- S výjimkou produktu WebSphere MQ for z/OSlze bez ohledu na to, zda jsou úplné, lze načítat pouze logické zprávy s možností načtení.
- Zda se zprávy ve skupině mohou načítat pouze tehdy, když jsou k dispozici *všechny* zprávy ve skupině
- S výjimkou produktu WebSphere MQ for z/OSlze segmenty v logické zprávě načítat pouze v případě, že jsou k dispozici *všechny* segmenty v logické zprávě.

Pokud ponecháte pole *Options* nastaveno na výchozí hodnotu (MQGMO_NO_WAIT), volání MQGET funguje tímto způsobem:

- Pokud neexistuje žádná zpráva odpovídající kritériím výběru ve frontě, volání nebude čekat na příchod zprávy, ale provede se okamžitě. Také v produktu WebSphere MQ for z/OS, volání nenastaví upozornění požadující oznámení, když taková zpráva dorazí.
- Způsob, jakým volání pracuje se synchronizačních bodů, je určen platformou:

Platforma	**Pod ovládacím prvkem synchronizačního bodu**
IBM i	Ne

Platforma	Pod ovládacím prvkem synchronizačního bodu
Systémy UNIX and Linux	Ne
z/OS	Ano
Systémy Windows	Ne

- V produktu WebSphere MQ for z/OSse načtená zpráva neoznačí jako přeskočení odvolání.
- Vybraná zpráva bude odebrána z fronty (není procházena).
- Nepožaduje se žádná konverze dat zprávy aplikace.
- Volání selže, pokud je zpráva delší než vaše vyrovnávací paměť.

WaitInterval
Pole *WaitInterval* uvádí maximální dobu (v milisekundách), po kterou volání MQGET čeká na příchod zprávy do fronty, když použijete volbu MQGMO_WAIT. Pokud do doby uvedené v souboru *WaitInterval*nepřijde žádná zpráva, volání se dokončí a vrátí kód příčiny, který ukazuje, že nebyla nalezena žádná zpráva, která by odpovídala kritériím výběru ve frontě.

Pokud v produktu WebSphere MQ for z/OSpoužijete volbu MQGMO_SET_SIGNAL, určuje pole *WaitInterval* čas, pro který je nastaven signál.

Další informace o těchto volbách viz “Čekání na zprávy” na stránce 256.

Signal1
HodnotaSignal1 je podporována pouze v produktu WebSphere MQ for z/OS a MQSeries pouze pro produkt HP Integrity NonStop Server.

Pokud použijete volbu MQGMO_SET_SIGNAL k požadavku, aby byla vaše aplikace oznámena při doručení vhodné zprávy, určete typ signálu v poli *Signal1* . V produktu WebSphere MQ na všech ostatních platformách je pole *Signal1* vyhrazené a jeho hodnota je nevýznamná.

Signal2
Pole *Signal2* je vyhrazeno na všech platformách a jeho hodnota není významná.

ResolvedQName
ResolvedQName je výstupní pole, v němž správce front vrací název fronty (po vyřešení jakéhokoli aliasu), ze kterého byla zpráva načtena.

MatchOptions
Produkt *MatchOptions* řídí kritéria výběru pro příkaz MQGET.

GroupStatus
GroupStatus udává, zda je zpráva, kterou jste načetli, ve skupině.

SegmentStatus
SegmentStatus udává, zda položka, kterou jste načetli, je segmentem logické zprávy.

Segmentation
Segmentation udává, zda je segmentace povolena pro načtenou zprávu.

MsgToken

MsgToken Jedinečně identifikuje zprávu.

ReturnedLength
ReturnedLength je výstupní pole, v němž správce front vrací délku vrácených dat zpráv (v bajtech).

MsgHandle
Manipulátor se zprávou, která má být naplněna vlastnostmi zprávy načítané z fronty. Popisovač byl dříve vytvořen voláním MQCRTMH. Všechny vlastnosti, které jsou již přidruženy k popisovači, jsou před načtením zprávy vymazány.

Určení velikosti oblasti vyrovnávací paměti

V argumentu *BufferLength* příkazu MQGET zadejte velikost oblasti vyrovnávací paměti, která má obsahovat data zpráv, která načítáte. Vy rozhodujete, jak velká by měla být ve třech směrech:

1. Možná již víte, jakou délku zpráv očekáváte od tohoto programu. Je-li tomu tak, uveďte vyrovnávací paměť této velikosti.

   Můžete však použít volbu MQGMO_ACCEPT_TRUNCATED_MSG ve struktuře MQGMO, pokud má být volání MQGET dokončeno i v případě, že je zpráva příliš velká pro vyrovnávací paměť. V tomto případě:

   - Vyrovnávací paměť je zaplněna jako velká část zprávy, jak ji lze zadržet.
   - Volání vrátí kód dokončení varování.
   - Zpráva se odstraní z fronty (zruší zbývající část zprávy) nebo je kurzor procházení zálohován (pokud procházíte frontu)
   - Skutečná délka zprávy je vrácena v produktu *DataLength*

   Bez této volby je volání stále dokončeno s varováním, ale neodebere zprávu z fronty (nebo zálohuje kurzor procházení).
2. Odhadněte velikost vyrovnávací paměti (nebo dokonce určete velikost nula bajtů) a *nepoužívat* volbu MQGMO_ACCEPT_TRUNCATED_MSG. Pokud se volání MQGET nezdaří (například proto, že vyrovnávací paměť je příliš malá), je délka zprávy vrácena v parametru *DataLength* volání. (Vyrovnávací paměť je stále zaplněna jako velká část zprávy, jak ji lze zadržet, ale zpracování volání není dokončeno.) Uložte *MsgId* této zprávy, pak zopakujte volání MQGET, uvedení oblasti vyrovnávací paměti o správné velikosti a *MsgId* , které jste si poznamenali z prvního volání.

   Pokud váš program obsluhuje frontu, která je také obsluhována jinými programy, může jeden z těchto jiných programů odstranit zprávu, kterou chcete, než bude moci váš program vydat další volání MQGET. Váš program může ztrácet čas hledáním zprávy, která již neexistuje. Chcete-li se tomu vyhnout, nejprve projděte frontu, dokud nenaleznete požadovanou zprávu, zadáním *BufferLength* hodnoty nula a pomocí volby MQGMO_ACCEPT_TRUNCATED_MSG. Tato pozice umístí kurzor pod zprávu, kterou chcete. Poté můžete znovu načíst zprávu vyvoláním příkazu MQGET a určením volby MQGMO_MSG_UNDER_CURSOR. Pokud jiný program odstraní zprávu mezi voláními pro procházení a odebrání, váš druhý příkaz MQGET selže ihned (bez prohledání celé fronty), protože pod vaším kurzorem procházení není žádná zpráva.
3. Atribut *MaxMsgLength queue* určuje maximální délku zpráv přijatých pro tuto frontu; atribut *MaxMsgLength správce front* určuje maximální délku zpráv přijatých pro daného správce front. Pokud nevíte, jakou délku zprávy očekávat, můžete se dotázat na atribut *MaxMsgLength* (pomocí volání MQINQ), pak zadat vyrovnávací paměť této velikosti.

   Pokuste se velikost vyrovnávací paměti co nejblíže ke skutečné velikosti zprávy, aby nedošlo ke snížení výkonu.

   Další informace o atributu *MaxMsgLength* viz “Zvýšení maximální délky zprávy” na stránce 251.

Pořadí, ve kterém jsou zprávy načítány z fronty

Pořadí, ve kterém načítáte zprávy z fronty, můžete řídit. Tato sekce se zaměřuje na volby.

Priorita

Program může při vložení zprávy do fronty přiřadit prioritu zprávy (viz “Priority zpráv” na stránce 17). Zprávy se stejnou prioritou jsou uloženy ve frontě v pořadí příchodu, nikoli pořadí, v jakém jsou potvrzeny.

Správce front udržuje fronty buď ve striktní posloupnosti FIFO (první dovnitř, první ven), nebo ve FIFO v rámci posloupnosti priority. Závisí to na nastavení atributu *MsgDeliverySequence* fronty.Když zpráva dorazí do fronty, je vložena okamžitě za poslední zprávou, která má stejnou prioritu.

Programy mohou buď získat první zprávu z fronty, nebo mohou získat určitou zprávu z fronty a ignorovat prioritu těchto zpráv. Program může například chtít zpracovat odpověď na konkrétní zprávu, kterou odeslal dříve. Další informace viz “Získání konkrétní zprávy” na stránce 246.

Pokud aplikace vkládá posloupnost zpráv do fronty, může další aplikace tyto zprávy načíst ve stejném pořadí, ve kterém byly vloženy, za předpokladu, že:

- Všechny zprávy mají stejnou prioritu
- Všechny zprávy byly vloženy do stejné jednotky práce nebo byly všechny vloženy mimo jednotku práce.
- Fronta je lokální vzhledem k aplikaci vkládání

Pokud tyto podmínky nejsou splněny a aplikace závisí na zprávách, které jsou načítány v určitém pořadí, aplikace musí buď zahrnout informace o posloupnosti do dat zprávy, nebo vytvořit prostředek pro potvrzení přijetí zprávy před odesláním další zprávy.

Logické a fyzické uspořádání

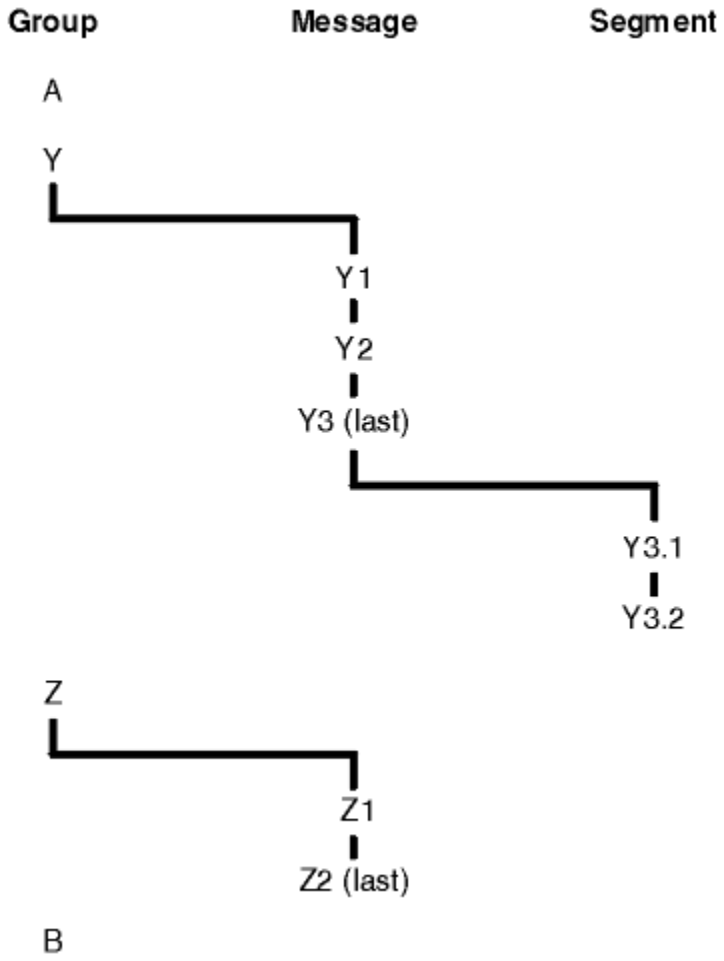
Zprávy ve frontách se mohou vyskytnout (v rámci každé úrovně priority) ve *fyzickém* nebo *logickém* pořadí.

Fyzické pořadí je pořadí, ve kterém zprávy dorazí do fronty. Logické pořadí je, když všechny zprávy a segmenty uvnitř skupiny jsou ve své logické posloupnosti, vedle sebe, v pozici určené fyzickou pozicí první položky patřící do skupiny.

Popis skupin, zpráv a segmentů najdete v tématu “Skupiny zpráv” na stránce 34. Tyto fyzické a logické pořadí se mohou lišit, protože:

- Skupiny mohou přijít na místo určení v podobných časech z různých aplikací, takže ztratí jakékoli odlišné fyzické pořadí.
- I v rámci jedné skupiny se mohou zprávy dostat mimo pořadí, protože došlo k přesměrování nebo zpoždění některých zpráv ve skupině.

Například, logický příkaz může vypadat jako obrázek Obrázek 34 na stránce 236:

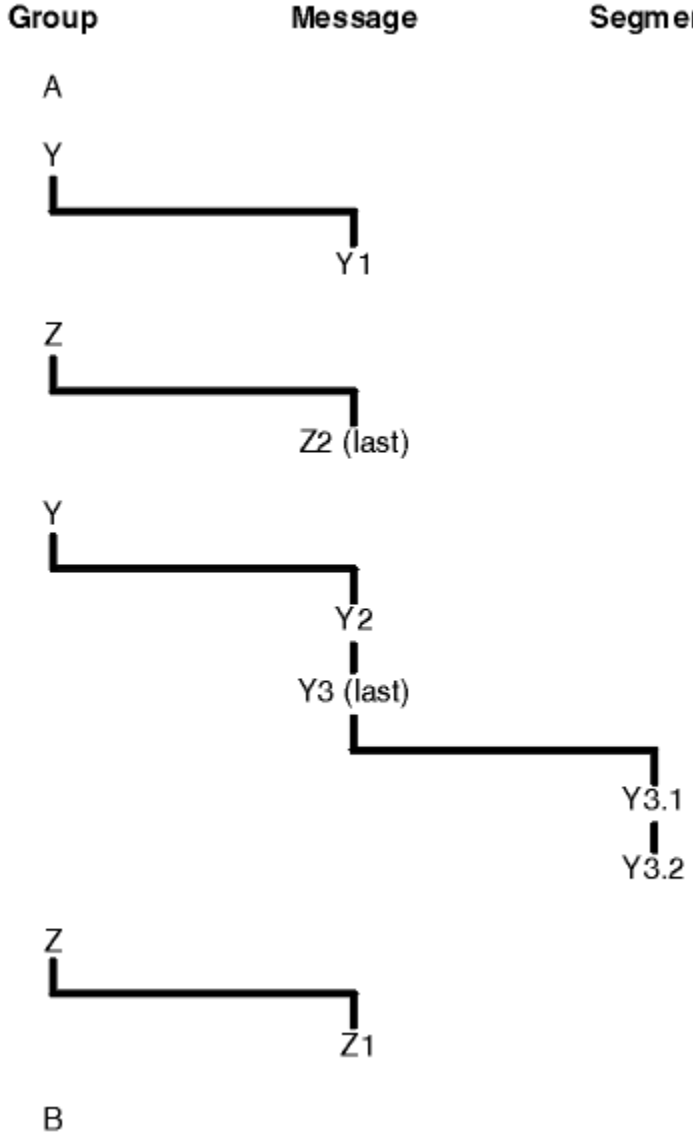


Obrázek 34. Logické pořadí ve frontě

Tyto zprávy se objeví v následujícím logickém pořadí ve frontě:

1. Zpráva A (ne ve skupině)
2. Logická zpráva 1 skupiny Y
3. Logická zpráva 2 skupiny Y
4. Segment 1 z (poslední) logické zprávy 3 skupiny Y
5. (Poslední) segment 2 z (poslední) logické zprávy 3 skupiny Y
6. Logická zpráva 1 skupiny Z
7. (Poslední) logická zpráva 2 skupiny Z
8. Zpráva B (ne ve skupině)

Fyzické uspořádání by však mohlo být zcela odlišné. Fyzická pozice *první* položky v každé skupině určuje logickou pozici celé skupiny. Například, pokud skupiny Y a Z dorazili v podobném čase a zpráva 2 skupiny Z převzala zprávu 1 stejné skupiny, fyzický příkaz by vypadal jako obrázek Obrázek 35 na stránce 237:



Obrázek 35. Fyzické pořadí ve frontě

Tyto zprávy se objevují v následujícím fyzickém pořadí na frontě:

1. Zpráva A (ne ve skupině)
2. Logická zpráva 1 skupiny Y

3. Logická zpráva 2 skupiny Z
4. Logická zpráva 2 skupiny Y
5. Segment 1 z (poslední) logické zprávy 3 skupiny Y
6. (Poslední) segment 2 z (poslední) logické zprávy 3 skupiny Y
7. Logická zpráva 1 skupiny Z
8. Zpráva B (ne ve skupině)

Poznámka: V systému IBM WebSphere MQ for z/OSnení fyzické pořadí zpráv ve frontě zaručeno, je-li fronta indexována pomocí GROUPID.

Při získávání zpráv můžete zadat MQGMO_LOGICAL_ORDER, chcete-li načítat zprávy v logickém pořadí spíše než ve fyzickém pořadí.

Zadáte-li volání MQGET s klauzulí MQGMO_BRON FIRST a MQGMO_LOGICAL_ORDER, musí následné volání MQGET s MQGMO_BROD NEXT také určovat MQGMO_LOGICAL_ORDER. Naopak, pokud MQGET s MQGMO_BROE_FIRST neurčuje MQGMO_LOGICAL_ORDER, nesmí ani následující MQGET s MQGMO_BRONEM NEXT.

Informace o skupinách a segmentech, které správce front uchovává pro volání MQGET, která procházejí zprávy ve frontě, je oddělena od informací o skupině a segmentu, které správce front uchovává pro volání MQGET, která odebírá zprávy z fronty. Když uvedete MQGMO_BE_FIRST, správce front ignoruje informace o skupině a segmentu pro procházení a prohledá frontu, jako by neexistovala žádná aktuální skupina a žádná aktuální logická zpráva.

Poznámka: Nepoužívejte volání MQGET k procházení *za koncem* skupiny zpráv (nebo logické zprávy, která se nenachází ve skupině) bez určení hodnoty MQGMO_LOGICAL_ORDER. Například, pokud poslední zpráva ve skupině *předchází* první zprávě ve skupině ve frontě, pomocí MQGMO_BERE NEXT k procházení za koncem skupiny, uvedení MQMO_MATCH_MSG_SEQ_NUMBER s `MsgSeqNumber` nastaveným na 1 (vyhledat první zprávu další skupiny) vrátí znovu první zprávu ve skupině již prohlédnuto. K tomu může dojít okamžitě nebo k několika dalším voláním MQGET (pokud jsou mezi nimi nějaké vedlejší skupiny).

Zamezte možnosti nekonečné smyčky otevřením fronty *dvakrát* pro procházení:

- Použijte první popisovač k procházení pouze první zprávy v každé skupině.
- Druhý ovladač použijte k procházení pouze zpráv v rámci určité skupiny.
- Použijte volby MQMO_ * k přesunutí druhého kurzoru pro procházení na pozici prvního kurzoru pro procházení, a teprve pak můžete procházet zprávy ve skupině.
- Nepoužívejte volbu MQGMO_BROWSE NEXT, než je konec skupiny.

Další informace naleznete v tématu MQGET, MQMDa Pravidla pro ověřování platnosti voleb MQI.

U většiny aplikací si při procházení pravděpodobně vyberete logické nebo fyzické uspořádání. Pokud však chcete přepínat mezi těmito režimy, pamatujte na to, že když nejprve zadáte příkaz k procházení MQGMO_LOGICAL_ORDER, bude vaše pozice v rámci logické posloupnosti vytvořena.

Pokud první položka v rámci skupiny není momentálně přítomna, skupina, kterou jste v této době, není považována za součást logické posloupnosti.

Jakmile se kurzor procházení nachází uvnitř skupiny, může pokračovat ve stejné skupině, i když je první zpráva odebrána. Na počátku se však nikdy nemůžete přesunout do skupiny pomocí MQGMO_LOGICAL_ORDER, kde první položka není přítomna.

MQPMO_LOGICAL_ORDER
Volba MQPMO sděluje správci front, jak aplikace vkládá zprávy do skupin a segmentů logických zpráv. Může být zadán pouze na volání MQPUT; není platný na volání MQPUT1 .

Je-li zadán parametr MQPMO_LOGICAL_ORDER, znamená to, že aplikace bude používat následná volání MQPUT, aby:

1. Vložila segmenty do každé logické zprávy kvůli zvýšení offsetu segmentu, počínaje 0, bez mezer.
2. Vložila všechny segmenty do jedné logické zprávy, a teprve pak vložila segment do další logické zprávy.

3. Vložila logické zprávy do každé skupiny zprávy, aby zvýšila pořadové číslo zprávy, počínaje 1, bez mezer.IBM WebSphere MQ automaticky zvýší pořadové číslo zprávy.
4. Vložila všechny logické zprávy do jedné skupiny zpráv, a teprve pak vložila logické zprávy do další skupiny zpráv.

Vzhledem k tomu, že aplikace sdělila správci front, jak vkládá zprávy do skupin a segmentů logických zpráv, aplikace nemusí udržovat a aktualizovat informace o skupinách a segmentech o každém volání MQPUT, protože správce front je uchovává a aktualizuje. Konkrétně to znamená, že aplikace nemusí nastavovat pole *GroupId*, *MsgSeqNumber*a *Offset* v MQMD, protože správce front nastavuje tato pole na příslušné hodnoty. Aplikace musí v produktu MQMD nastavit pouze pole *MsgFlags* , aby označovala, kdy zprávy patří do skupin nebo jsou segmenty logických zpráv, a označují poslední zprávu ve skupině nebo posledním segmentu logické zprávy.

Po spuštění skupiny zpráv nebo logické zprávy musí následná volání MQPUT určovat příslušné příznaky MQMF_ * v produktu *MsgFlags* v deskriptoru MQMD. Pokud se aplikace pokusí vložit zprávu, která není ve skupině, když existuje neukončená skupina zpráv, nebo pokud se jedná o neukončenou logickou zprávu, která není segmentem, volání selže s kódem příčiny MQRC_INCOMPLEE_GROUP nebo MQRC_INCOMPLEE_MSG, jak je to vhodné. Správce front však uchovává informace o aktuální skupině zpráv nebo aktuální logické zprávě a aplikace ji může ukončit odesláním zprávy (případně bez dat zprávy aplikace) zadáním volání MQMF_LAST_MSG_IN_GROUP nebo MQMF_LAST_SEGMENT před opětným zadáním volání MQPUT pro vložení zprávy, která není ve skupině, nebo se nejedná o segment.

Příkaz Obrázek 35 na stránce 237 zobrazuje kombinace voleb a příznaků, které jsou platné, a hodnoty polí *GroupId*, *MsgSeqNumber*a *Offset* , které správce front používá v každém případě. Kombinace voleb a příznaků, které nejsou zobrazeny v tabulce, jsou neplatné. Sloupce v tabulce mají následující významy: Either means Yes or No No:

LOG SLOVO
Určuje, zda je v rámci volání zadána volba MQPMO_LOGICAL_ORDER.

MIG
Určuje, zda je v rámci volání zadána volba MQMF_MSG_IN_GROUP nebo MQMF_LAST_IN_GROUP.

SEG
Určuje, zda je v rámci volání zadána volba MQMF_SEGMENT nebo MQMF_LAST_SEGMENT.

SEG OK
Určuje, zda je u volání zadána volba MQMF_SEGMENTATION_ALLOWED.

Cur grp
Určuje, zda před voláním existuje aktuální skupina zpráv.

Zpráva Cur msg
Určuje, zda před voláním existuje aktuální logická zpráva.

Ostatní sloupce
Zobrazení hodnot, které správce front používá. Předchozí označuje hodnotu použitou pro pole v předchozí zprávě pro popisovač fronty.

Tabulka 36. Volby MQPUT související se zprávami ve skupinách a segmentech logických zpráv

Volby, které uvedete				**Stav skupiny a protokolu- zpráva před voláním**		**Hodnoty, které správce front používá**		
PROTOKOL	**MIGOCITY**	**SEGG**	**SEG OK**	**Cur grp**	**Zpráva protokolu cur**	***GroupId***	***MsgSeqNumber***	***Offset***
Ano	Ne	Ne	Ne	Ne	Ne	MQGI_NONE	1	0

Tabulka 36. Volby MQPUT související se zprávami ve skupinách a segmentech logických zpráv (pokračování)

Volby, které uvedete				Stav skupiny a protokolu-zpráva před voláním		Hodnoty, které správce front používá		
PROTOKOL	**MIGOCITY**	**SEGG**	**SEGOK**	**Cur grp**	**Zpráva protokolu cur**	***GroupId***	***MsgSeqNumber***	***Offset***
Ano	Ne	Ne	Ano	Ne	Ne	ID nové skupiny	1	0
Ano	Ne	Ano	buď	Ne	Ne	ID nové skupiny	1	0
Ano	Ne	Ano	buď	Ne	Ano	Předchozí ID skupiny	1	Předchozí odchylka + předchozí délka segmentu
Ano	Ano	buď	buď	Ne	Ne	ID nové skupiny	1	0
Ano	Ano	buď	buď	Ano	Ne	Předchozí ID skupiny	Předchozí pořadové číslo + 1	0
Ano	Ano	Ano	buď	Ano	Ano	Předchozí ID skupiny	Předchozí pořadové číslo	Předchozí odchylka + předchozí délka segmentu
Ne	Ne	Ne	Ne	buď	buď	MQGI_NONE	1	0
Ne	Ne	Ne	Ano	buď	buď	Nové ID skupiny, je-li hodnota MQGI_NONE, další hodnota v poli	1	0
Ne	Ne	Ano	buď	buď	buď	Nové ID skupiny, je-li hodnota MQGI_NONE, další hodnota v poli	1	Hodnota v poli
Ne	Ano	Ne	buď	buď	buď	Nové ID skupiny, je-li hodnota MQGI_NONE, další hodnota v poli	Hodnota v poli	0
Ne	Ano	Ano	buď	buď	buď	Nové ID skupiny, je-li hodnota MQGI_NONE, další hodnota v poli	Hodnota v poli	Hodnota v poli

Poznámka:

- Volání MQPMO_LOGICAL_ORDER není v rámci volání MQPUT1 platné.
- Pro pole *MsgId* správce front generuje nový identifikátor zprávy, je-li zadáno MQPMO_NEW_MSG_ID nebo MQMI_NONE, a v opačném případě použije hodnotu v poli.
- Pro pole *CorrelId* správce front vygeneruje nový korelační identifikátor, pokud je zadán parametr MQPMO_NEW_CORREL_ID a v opačném případě použije hodnotu v poli.

Určíte-li MQPMO_LOGICAL_ORDER, správce front vyžaduje, aby všechny zprávy ve skupině a segmenty v logické zprávě byly vloženy se stejnou hodnotou do pole *Persistence* v MQMD, tj. všechny musí být trvalé nebo všechny musí být přechodné. Není-li tato podmínka splněna, volání MQPUT selže s kódem příčiny MQRC_INCONSISTENT_PERSISTENCE.

Volba MQPMO_LOGICAL_ORDER má vliv na jednotky práce následujícím způsobem:

- Je-li první fyzická zpráva ve skupině nebo logické zprávě vložena do pracovní jednotky, musí být všechny ostatní fyzické zprávy ve skupině nebo v logické zprávě vloženy do jednotky práce, pokud se použije stejný popisovač fronty. Nepotřebují však být vloženy do stejné pracovní jednotky, což umožňuje skupině zpráv nebo logické zprávě, která se skládá z mnoha fyzických zpráv, které mají být rozděleny do dvou nebo více po sobě jdoucích jednotek práce pro manipulátor fronty.
- Pokud se první fyzická zpráva ve skupině nebo logické zprávě nevloží do pracovní jednotky, žádná z jiných fyzických zpráv ve skupině nebo logické zprávě nemůže být vložena do pracovní jednotky, pokud se použije stejný popisovač fronty.

Nejsou-li tyto podmínky splněny, volání MQPUT selže s kódem příčiny MQRC_INCONSISTENT_UOW.

Je-li zadán parametr MQPMO_LOGICAL_ORDER, MQMD zadaný v rámci volání MQPUT nesmí být menší než hodnota MQMD_VERSION_2. Není-li tato podmínka splněna, volání selže s kódem příčiny MQRC_WRONG_MD_VERSION.

Není-li uvedeno MQPMO_LOGICAL_ORDER, zprávy ve skupinách a segmentech logických zpráv lze vložit do libovolného pořadí a není nutné vkládat úplné skupiny zpráv ani úplné logické zprávy. Je odpovědností aplikace, aby zajistila, že pole *GroupId*, *MsgSeqNumber*, *Offset*a *MsgFlags* mají odpovídající hodnoty.

Tuto techniku použijte k restartování skupiny zpráv nebo logické zprávy uprostřed poté, co došlo k selhání systému. Když se systém restartuje, může aplikace nastavit pole *GroupId*, *MsgSeqNumber*, *Offset*, *MsgFlags*a *Persistence* na příslušné hodnoty a poté vydat volání MQPUT s parametrem MQPMO_SYNCPOINT nebo MQPMO_NO_SYNCPOINT, jak je požadováno, ale bez určení hodnoty MQPMO_LOGICAL_ORDER. Je-li toto volání úspěšné, uchovává správce front informace o skupině a segmentu a následná volání MQPUT používající tento manipulátor fronty mohou jako normální určit MQPMO_LOGICAL_ORDER.

Informace o skupinách a segmentech, které správce front uchovává pro volání MQPUT, jsou odděleny od informací o skupině a segmentu, které si zachovává pro volání MQGET.

Pro daný popisovač fronty může aplikace směšovat volání MQPUT, která určuje volání MQPMO_LOGICAL_ORDER s voláními MQPUT, ale povšimněte si následujících bodů:

- Není-li parametr MQPMO_LOGICAL_ORDER zadán, způsobí každé úspěšné volání MQPUT správce front tak, aby nastavil informace o skupině a segmentu pro manipulátor fronty na hodnoty zadané aplikací a nahradí existující informace o skupině a segmentech zachované správcem front pro manipulátor fronty.
- Není-li parametr MQPMO_LOGICAL_ORDER zadán, volání se nezdaří, pokud existuje aktuální skupina zpráv nebo logická zpráva; volání může být úspěšné s kódem dokončení MQCC_WARNING. Tabulka 37 na stránce 242 ukazuje různé případy, které mohou nastat. V těchto případech, pokud kód dokončení není MQCC_OK, je kód příčiny jedním z následujících (je-li to vhodné):
  - SKUPINA MQRC_INCOMPLETE_GROUP
  - ZPRÁVA MQRC_INCOMPLETE_MSG
  - MQRC_INCONSISTENT_PERSISTENCE
  - NEKONZISTENCE MQRC_INCONSISTENT_UOW

  Poznámka: Správce front nekontroluje informace o skupině a segmentu pro volání MQPUT1 .

Tabulka 37. Výsledek, když volání MQPUT nebo MQCLOSE není konzistentní s informacemi o skupině a segmentu		
Aktuální volání je	**Předchozí volání bylo MQPUT s MQPMO_LOGICAL_ORDER**	**Předchozí volání bylo MQPUT bez MQPMO_LOGICAL_ORDER**
MQPUT s MQPMO_LOGICAL_ORDER	SELHÁNÍ MQCC_FAILED	SELHÁNÍ MQCC_FAILED
MQPUT bez MQPMO_LOGICAL_ORDER	VAROVÁNÍ MQCC_WARNING	MQCC_OK
MQCLOSE s neukončené skupinou nebo logickou zprávou	VAROVÁNÍ MQCC_WARNING	MQCC_OK

Pro aplikace, které vložila zprávy a segmenty v logickém pořadí, zadejte MQPMO_LOGICAL_ORDER, protože je to nejjednodušší volba, která se má použít. Tato volba zbavuje aplikaci potřeby spravovat informace o skupinách a segmentech, protože tyto informace spravuje správce front. Nicméně specializované aplikace mohou vyžadovat větší kontrolu nad možností volby MQPMO_LOGICAL_ORDER, čehož lze dosáhnout neurčením této volby; pokud tak učiníte, musíte zajistit, aby pole*GroupId*, *MsgSeqNumber*, *Offset*a *MsgFlags* v MQMD byly nastaveny správně, a to před každým voláním MQPUT nebo MQPUT1 .

Například aplikace, která chce předat fyzickým zprávám, které přijímá, bez ohledu na to, zda se tyto zprávy nacházejí ve skupinách nebo segmentech logických zpráv, nesmí určovat MQPMO_LOGICAL_ORDER, a to ze dvou důvodů:

- Pokud jsou zprávy načteny a uvedeny v pořadí, určuje parametr MQPMO_LOGICAL_ORDER nový identifikátor skupiny pro zprávy, což může způsobit, že odesilateli zpráv může být obtížné nebo nemožné korelovat všechny zprávy odpovědi nebo zprávy, které jsou výsledkem skupiny zpráv.
- Ve složité síti s více cestami mezi odesílajícím a přijímajícím správcem front může dojít k nedostatku fyzických zpráv v pořadí. Neuvedení MQPMO_LOGICAL_ORDER a MQGMO_LOGICAL_ORDER na volání MQGET může pro každou fyzickou zprávu načíst a předat každou fyzickou zprávu, jakmile dorazí, aniž by čekal na příchod dalšího v logickém pořadí, aby se dospělo.

Aplikace, které generují zprávy sestav pro zprávy ve skupinách nebo segmentech logických zpráv, nesmí při vkládání zprávy sestavy uvádět také MQPMO_LOGICAL_ORDER.

MQPMO_LOGICAL_ORDER lze zadat s libovolněmi dalšími volbami MQPMO_ *.

Umístění logicky uspořádaných skupin do klastrované fronty (MQOO_BIND_ON_GROUP)

Volba MQOO_BIND_ON_OPEN zajišťuje, že všechny zprávy z této aplikace, a tedy všechny skupiny, jsou směrovány do jediné instance. To má nevýhodu, že provoz aplikací není vyrovnán přes více instancí fronty klastru. Chcete-li povolit vyrovnávání pracovní zátěže při zachování nedotčených skupin zpráv, je třeba nastavit následující volby:

- Volání MQPUT musí určovat MQPMO_LOGICAL_ORDER.
- Volání MQOPEN musí uvádět jednu z následujících dvou voleb:
  - SKUPINA MQO_BIND_ON_GROUP
  - MQOO_BIND_AS_Q_DEF a definice fronty musí určovat DEFBIND (GROUP)

Vyrovnávání pracovní zátěže je poté řízeno *mezi skupinami* zpráv bez nutnosti použití příkazů MQCLOSE a MQOPEN fronty. *Mezi skupinami* znamená, že proměnná MQMF_MSG_IN_GROUP je nastavena v deskriptoru MQMD (v2) nebo MQMDE a ve zpracování neexistuje žádná částečně dokončená skupina. Když probíhá skupina, budou vyřešený správce front a název fronty v manipulátoru objektu použity znovu.

Pokud byla předchozí zpráva nastavena na hodnotu MQPMO_LOGICAL_ORDER nebo MQMF_MSG_IN_GROUP, ale aktuální zpráva není součástí skupiny, volání PUT selže s hodnotou MQRC_INCOMPLEE_GROUP.

Pokud pro jednotlivé příkazy MQPUT není určen parametr MQPMO_LOGICAL_ORDER a žádná aktuální skupina není aktivní, bude pro tuto zprávu provedeno vyrovnávání pracovní zátěže (jako by volání MQOPEN určoval MQOO_BIND_NOT_FIXED).

Pro zprávy vázané na místo určení pomocí struktury MQOO_BIND_ON_GROUP není provedeno žádné opětovné přidělení. Další informace o opětovném přidělení naleznete v tématu “Skupiny zpráv” na stránce 34.

Seskupení logických zpráv

Pro použití logických zpráv ve skupině existují dva hlavní důvody:

- Je možné, že bude třeba zprávy zpracovat v určitém pořadí.
- Možná budete muset každou zprávu ve skupině zpracovat v souvisejícím způsobem.

V obou případech načtěte celou skupinu se stejnou instancí aplikace pro získání aplikace.

Předpokládejme například, že skupina se skládá ze čtyř logických zpráv. Aplikace bude vypadat takto:

```
PMO.Options = MQPMO_LOGICAL_ORDER | MQPMO_SYNCPOINT

MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
MQPUT MD.MsgFlags = MQMF_LAST_MSG_IN_GROUP

MQCMIT
```

Aplikace pro získání určuje volbu MQGMO_ALL_MSGS_AVAILABLE pro první zprávu ve skupině. Tím je zajištěno, že zpracování se nespustí, dokud nedorazíte všechny zprávy v rámci skupiny. Volba MQGMO_ALL_MSGS_AVAILABLE je ignorována pro následné zprávy ve skupině.

Když je načtena první logická zpráva skupiny, můžete použít příkaz MQGMO_LOGICAL_ORDER, abyste se ujistili, že zbývající logické zprávy skupiny jsou načteny v pořadí.

Takže aplikace pro získání vypadá takto:

```
/* Wait for the first message in a group, or a message not in a group */
GMO.Options = MQGMO_SYNCPOINT | MQGMO_WAIT
            | MQGMO_ALL_MSGS_AVAILABLE | MQGMO_LOGICAL_ORDER
do while ( GroupStatus == MQGS_MSG_IN_GROUP )
   MQGET
   /* Process each remaining message in the group */
   ...

MQCMIT
```

Další příklady seskupování zpráv viz “Segmentace aplikace logických zpráv” na stránce 254 a “Uvedení a získání skupiny, která zahrnuje jednotky práce” na stránce 243.

Informace o povolení požadavku na to, aby skupina zpráv byla alokována do stejné cílové instance pro fronty klastru, naleznete v části DefBind.

Uvedení a získání skupiny, která zahrnuje jednotky práce

V předchozím případě nemohou zprávy nebo segmenty začínat na opuštění uzlu (je-li jeho cíl vzdálený) nebo se má začít načítat, dokud není vložena celá skupina a jednotka práce je potvrzená. To nemusí být to, co chcete, pokud trvá dlouho, než se celá skupina, nebo je-li prostor fronty na uzlu omezený. K vyřešení tohoto stavu, dát skupinu do několika jednotek práce.

Je-li skupina vložena do více jednotek práce, je možné, aby se některé skupiny zavázaly, i když se aplikace nezdaří. Aplikace proto musí ukládat informace o stavu do každé jednotky práce, kterou může použít po restartování, aby mohla být obnovena nekompletní skupina. Nejjednodušší místo pro záznam těchto informací je ve frontě STATUS. Pokud byla úspěšně vložena úplná skupina, fronta STATUS je prázdná.

Je-li zahrnuta segmentace, logika je podobná. V takovém případě musí příkaz StatusInfo obsahovat *Offset* .

Zde je příklad uvedení skupiny do několika jednotek práce:

```
PMO.Options = MQPMO_LOGICAL_ORDER | MQPMO_SYNCPOINT

/* First UOW */

MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
StatusInfo = GroupId,MsgSeqNumber from MQMD
MQPUT (StatusInfo to STATUS queue) PMO.Options = MQPMO_SYNCPOINT
MQCMIT

/* Next and subsequent UOWs */
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
MQGET (from STATUS queue) GMO.Options = MQGMO_SYNCPOINT
StatusInfo = GroupId,MsgSeqNumber from MQMD
MQPUT (StatusInfo to STATUS queue) PMO.Options = MQPMO_SYNCPOINT
MQCMIT

/* Last UOW */
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
MQPUT MD.MsgFlags = MQMF_LAST_MSG_IN_GROUP
MQGET (from STATUS queue) GMO.Options = MQGMO_SYNCPOINT
MQCMIT
```

Pokud byly potvrzeny všechny jednotky práce, celá skupina byla úspěšně vložena a fronta STATUS je prázdná. Pokud tomu tak není, skupina musí být znovu zahájena v bodě indikovaném informacemi o stavu. Objekt MQPMO_LOGICAL_ORDER nelze použít pro první vložení, ale poté jej lze použít.

Restart zpracování vypadá takto:

```
MQGET (StatusInfo from STATUS queue) GMO.Options = MQGMO_SYNCPOINT
if (Reason == MQRC_NO_MSG_AVAILABLE)
   /* Proceed to normal processing */
   ...

else
   /* Group was terminated prematurely */
   Set GroupId, MsgSeqNumber in MQMD to values from Status message
   PMO.Options = MQPMO_SYNCPOINT
   MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP

   /* Now normal processing is resumed.
      Assume this is not the last message */
   PMO.Options = MQPMO_LOGICAL_ORDER | MQPMO_SYNCPOINT
   MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
   MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
   StatusInfo = GroupId,MsgSeqNumber from MQMD
   MQPUT (StatusInfo to STATUS queue) PMO.Options = MQPMO_SYNCPOINT
   MQCMIT
```

Od získání aplikace můžete začít zpracovávat zprávy ve skupině ještě před tím, než dorazila celá skupina. To zkracuje dobu odezvy ve zprávách v rámci skupiny, a také znamená, že úložiště není pro celou skupinu požadováno. Chcete-li si uvědomit výhody, použijte několik jednotek práce pro každou skupinu zpráv. Z důvodů zotavení je třeba načíst každou zprávu v rámci pracovní jednotky.

Stejně jako u příslušné aplikace vkládání je třeba zaznamenat informace o stavu, které mají být zaznamenávány automaticky, jakmile je každá jednotka práce potvrzená. Opět platí, že nejjednodušším místem pro záznam těchto informací je fronta STAVU. Pokud byla úplná skupina úspěšně zpracována, je fronta STATUS prázdná.

Poznámka: Pro intermediační jednotky práce se můžete vyhnout voláním MQGET z fronty STAVU tak, že každý z příkazů MQPUT do stavové fronty je segmentem zprávy (tj. nastavením příznaku MQMF_SEGMENT) místo toho, aby se pro každou jednotku práce zakládal úplná nová zpráva. V poslední pracovní jednotce je do stavové fronty vložen konečný segment s uvedením MQMF_LAST_SEGMENT a pak jsou informace o stavu vymazány s parametrem MQGET specifikací MQGMO_COMPLETE_MSG.

Během zpracování restartu místo použití jediné MQGET k získání možné stavové zprávy procházejte stavovou frontou s MQGMO_LOGICAL_ORDER, dokud nedosáhnete posledního segmentu (to znamená, dokud se nevrátí žádné další segmenty). V první pracovní jednotce po restartu také uveďte offset explicitně při umístění segmentu stavu.

V následujícím příkladu uvažujeme pouze o zprávách ve skupině za předpokladu, že vyrovnávací paměť aplikace je vždy dostatečně velká, aby udržela celou zprávu, ať už byla zpráva segmentována, či nikoli. MQGMO_COMPLETE_MSG je proto určen pro každý příkaz MQGET. Stejné zásady platí i pro segmentaci segmentace (v tomto případě musí StatusInfo obsahovat *Offset*).

Pro zjednodušení předpokládáme, že v rámci jediné jednotky UOW je načítána maximálně 4 zpráv:

```
msgs = 0    /* Counts messages retrieved within UOW */
/* Should be no status message at this point */

/* Retrieve remaining messages in the group */
do while ( GroupStatus == MQGS_MSG_IN_GROUP )

   /* Process up to 4 messages in the group */
   GMO.Options = MQGMO_SYNCPOINT | MQGMO_WAIT
               | MQGMO_LOGICAL_ORDER
   do while ( (GroupStatus == MQGS_MSG_IN_GROUP) && (msgs < 4) )
      MQGET
      msgs = msgs + 1
      /* Process this message */
      ...
   /* end while

   /* Have retrieved last message or 4 messages  */
   /* Update status message if not last in group */
   MQGET (from STATUS queue) GMO.Options = MQGMO_SYNCPOINT
   if ( GroupStatus == MQGS_MSG_IN_GROUP )
      StatusInfo = GroupId,MsgSeqNumber from MQMD
      MQPUT (StatusInfo to STATUS queue) PMO.Options = MQPMO_SYNCPOINT
   MQCMIT
   msgs = 0
/* end while

if ( msgs > 0 )
   /* Come here if there was only 1 message in the group */
   MQCMIT
```

Pokud byly potvrzeny všechny jednotky práce, celá skupina byla úspěšně načtena a fronta STATUS je prázdná. Pokud tomu tak není, skupina musí být znovu zahájena v bodě indikovaném informacemi o stavu. MQGMO_LOGICAL_ORDER nelze použít pro první načtení, ale poté jej lze použít.

Restart zpracování vypadá takto:

```
MQGET (from STATUS queue) GMO.Options = MQGMO_SYNCPOINT
if (Reason == MQRC_NO_MSG_AVAILABLE)
   /* Proceed to normal processing */
   ...

else
   /* Group was terminated prematurely */
   /* The next message on the group must be retrieved by matching
      the sequence number and group id with those retrieved from the
      status information. */
   GMO.Options = MQGMO_COMPLETE_MSG | MQGMO_SYNCPOINT | MQGMO_WAIT
   MQGET GMO.MatchOptions = MQMO_MATCH_GROUP_ID | MQMO_MATCH_MSG_SEQ_NUMBER,
         MQMD.GroupId      = value from Status message,
         MQMD.MsgSeqNumber = value from Status message plus 1
   msgs = 1
   /* Process this message */
   ...

   /* Now normal processing is resumed */
   /* Retrieve remaining messages in the group */
   do while ( GroupStatus == MQGS_MSG_IN_GROUP )

      /* Process up to 4 messages in the group */
      GMO.Options = MQGMO_COMPLETE_MSG | MQGMO_SYNCPOINT | MQGMO_WAIT
                  | MQGMO_LOGICAL_ORDER
      do while ( (GroupStatus == MQGS_MSG_IN_GROUP) && (msgs < 4) )
```

```
            MQGET
            msgs = msgs + 1
            /* Process this message */
            ...

         /* Have retrieved last message or 4 messages  */
         /* Update status message if not last in group */
         MQGET (from STATUS queue) GMO.Options = MQGMO_SYNCPOINT
         if ( GroupStatus == MQGS_MSG_IN_GROUP )
            StatusInfo = GroupId,MsgSeqNumber from MQMD
            MQPUT (StatusInfo to STATUS queue) PMO.Options = MQPMO_SYNCPOINT
         MQCMIT
         msgs = 0
```

Získání konkrétní zprávy

Existuje celá řada způsobů, jak získat určitou zprávu z fronty. To jsou: výběr na MsgId a CorrelId, výběr na GroupId, MsgSeqČíslo a Posunutí a výběr na MsgToken. Řetězec výběru můžete použít také při otevření fronty.

Chcete-li získat určitou zprávu z fronty, použijte pole *MsgId* a *CorrelId* struktury MQMD. Aplikace však mohou tato pole explicitně nastavit, takže vámi zadané hodnoty nemusí identifikovat jedinečnou zprávu. Produkt Tabulka 38 na stránce 246 zobrazuje, jaká zpráva je načtena pro možná nastavení těchto polí. Tato pole jsou na vstupu ignorována, pokud určujete parametr MQGMO_MSG_UNDER_CURSOR v parametru *GetMsgOpts* volání MQGET.

Tabulka 38. Použití identifikátorů zpráv a korelací

Načtení ...	***MsgId***	***CorrelId***
První zpráva ve frontě	MQMI_NONE	MQCI_NONE
První zpráva, která odpovídá *MsgId*	Nenulový	MQCI_NONE
První zpráva, která odpovídá *CorrelId*	MQMI_NONE	Nenulový
První zpráva, která odpovídá produktu *MsgId* i *CorrelId*	Nenulový	Nenulový

V každém případě *první* znamená první zprávu splňující kritéria výběru (pokud není zadán parametr MQGMO_BRONEXT NEXT, znamená to, že se jedná o zprávu *další* v posloupnosti splňující kritéria výběru).

Při návratu nastaví volání MQGET pole *MsgId* a *CorrelId* na identifikátory zprávy a korelační identifikátory vrácené zprávy, pokud existují.

Nastavíte-li pole *Version* struktury MQMD na hodnotu 2, můžete použít pole *GroupId*, *MsgSeqNumbe*r a *Offset* . Tabulka 39 na stránce 246 Ukazuje, jaká zpráva je načtena pro možná nastavení těchto polí.

Tabulka 39. Použití identifikátoru skupiny

Načtení ...	**Volby shody**
První zpráva ve frontě	MQMO_NONE
První zpráva, která odpovídá *MsgId*	MQMO_MATCH_MSG_ID
První zpráva, která odpovídá *CorrelId*	MQMO_MATCH_CORREL_ID
První zpráva, která odpovídá *GroupId*	MQMO_MATCH_GROUP_ID
První zpráva, která odpovídá *MsgSeqNumber*	MQMO_MATCH_MSG_SEQ_NUMBER
První zpráva, která odpovídá *MsgToken*	MQMO_MATCH_MSG_TOKEN
První zpráva, která odpovídá *Offset*	MQMO_MATCH_OFSET

Tabulka 39. Použití identifikátoru skupiny (pokračování)

Načtení ...	Volby shody

Notes:

1. Hodnota MQMO_MATCH_XXX znamená, že pole *XXX* ve struktuře MQMD je nastaveno na hodnotu, která má být porovnána.
2. Příznaky MQMO lze použít v kombinaci. Příklad: MQMO_MATCH_GROUP_ID, MQMO_MATCH_MSG_SEQ_NUMBER a MQMO_MATCH_OFFSET lze použít společně k poskytnutí segmentu identifikovaného pomocí polí *GroupId*, *MsgSeqNumbe*r a *Offset* .
3. Určíte-li MQGMO_LOGICAL_ORDER, bude ovlivněna zpráva, kterou se pokoušíte načíst, protože tato volba závisí na informacích o stavu řízených pro manipulátor fronty. Další informace o tomto tématu viz “Logické a fyzické uspořádání” na stránce 236 a Volby.

Volání MQGET obvykle načte první zprávu z fronty. Zadáte-li konkrétní zprávu při použití volání MQGET, musí správce front hledat frontu, dokud nenajde příslušnou zprávu. To může ovlivnit výkon vaší aplikace.

Pokud používáte strukturu MQGMO verze 2 nebo novější a neurčujte příznaky MQMO_MATCH_MSG_ID nebo MQMO_MATCH_CORREL_ID, není třeba pole *MsgId* nebo *CorrelId* mezi příkazy MQGET resetovat.

Konkrétní zprávu z fronty můžete získat zadáním jeho hodnoty MsgToken a MatchOption MQMO_MATCH_MSG_TOKEN v rámci struktury MQGMO. Hodnota MsgToken je vrácena voláním MQPUT, které původně danou zprávu umístily do fronty, nebo předchozími operacemi MQGET a zůstává konstantní, dokud nebude správce front restartován.

Zajímáte-li se pouze o podmnožinu zpráv ve frontě, můžete určit, které zprávy chcete zpracovat, pomocí výběrového řetězce s voláním MQOPEN nebo MQSUB. Funkce MQGET poté načte další zprávu, která odpovídá tomuto řetězci výběru. Další informace o výběrových řetězcích najdete v tématu “Selektory.” na stránce 21.

Zlepšení výkonu přechodných zpráv

Když klient vyžaduje zprávu ze serveru, odešle požadavek na server. Odešle samostatný požadavek pro každou zprávu, kterou spotřebuje. Chcete-li zvýšit výkon klientů, kteří spotřebovávají přechodné zprávy, tím, že nechcete odesílat tyto zprávy požadavků, může být klient nakonfigurován tak, aby používal *čtení napřed*. Čtení napřed umožňuje odesílání zpráv klientovi, aniž by aplikace musela požadovat jejich zpracování.

Je-li povoleno čtení napřed, zprávy se posílají do vyrovnávací paměti na straně klienta s názvem *read ahead buffer*. Klient bude mít pro každou frontu dopředné vyrovnávací paměti čtení napřed s povoleným dopředným čtením. Zprávy ve vyrovnávací paměti dopředného čtení nejsou trvalé. Klient pravidelně aktualizuje server informacemi o množství dat, které spotřeboval.

Při volání MQOPEN s MQOH_READAHEAD klient WebSphere MQ povolí čtení napřed, pokud jsou splněny určité podmínky. Tyto podmínky zahrnují:

- Jak klient, tak i vzdálený správce front musí používat produkt WebSphere MQ verze 7 nebo novější.
- Klientská aplikace musí být zkompilována a propojena s knihovnami klienta WebSphere MQ MQI s podporou podprocesů.
- Kanál klienta musí používat protokol TCP/IP.
- Kanál musí mít nenulové nastavení SharingConversations (SHARECNV) v definici kanálu klienta i serveru.

Použití dopředného čtení může zlepšit výkon při spotřebovávání přechodných zpráv z klientské aplikace. Toto zlepšení výkonu je k dispozici pro aplikace MQI i JMS. Klientské aplikace používající příkaz MQGET nebo asynchronní spotřeba budou těžit ze zlepšení výkonu při přijímání přechodných zpráv.

Ne všechny návrhy aplikací klienta jsou vhodné pro použití čtení napřed, protože nejsou podporovány všechny volby pro použití s dopředným čtením a některé volby musí být konzistentní mezi voláními MQGET, je-li povoleno dopředné čtení. Pokud klient změní svá kritéria výběru mezi voláními MQGET,

zprávy uložené ve vyrovnávací paměti dopředného čtení zůstanou uvízlé v vyrovnávací paměti čtení napřed klienta.

Pokud již nejsou požadovány nevyřízené požadavky na uvízlé zprávy s předchozími výběrovými kritérii, lze na klientovi nastavit konfigurovatelný interval vymazání, aby se tyto zprávy od klienta automaticky vymazávaly. Interval vymazání je jedna ze skupiny voleb ladění dopředného čtení, kterou určuje klient. Tyto volby je možné vyladit tak, aby splňovaly vaše požadavky.

Je-li klientská aplikace restartována, mohou být zprávy v vyrovnávací paměti dopředného čtení ztraceny. A naopak, zpráva, která byla přesunuta do vyrovnávací paměti dopředného čtení, by mohla být odstraněna z podkladové fronty; nevzniká tím, že by byla z vyrovnávací paměti odebrána, takže volání MQGET s využitím dopředného čtení může vrátit zprávu, která již neexistuje.

Čtení napřed se provádí pouze pro vazby klienta. Atribut je ignorován pro všechny ostatní vazby.

Čtení napřed nemá žádný vliv na spuštění. Při dopředném čtení zprávy klientem není generována žádná zpráva spouštěče. Čtení napřed negeneruje informace evidence a statistiky, je-li povoleno.

Použití dopředného čtení s publikováním zasílání zpráv odběru

Když odběratelská aplikace určuje cílovou frontu, do které jsou publikace odesílána, použije se jako výchozí hodnota dopředného čtení hodnota DEFREADA zadané fronty.

Při požadavcích na odběry aplikací, které produkt WebSphere MQ spravuje místo určení, do kterého jsou publikování odesílána, je vytvořena spravovaná fronta jako dynamická fronta na základě předdefinované modelové fronty. Jedná se o hodnotu DEFREADA modelové fronty, která se používá jako výchozí hodnota dopředného čtení. Výchozí modelové fronty SYSTEM.DURABLE.PUBLICATIONS.MODEL nebo SYSTEM.NONDURABLE.PUBLICATIONS.MODEL se použije, pokud není definována modelová fronta pro toto nebo nadřízené téma.

Související pojmy

“Vyladění výkonu pro přechodné zprávy v produktu AIX” na stránce 250
Pokud používáte produkt AIX V5.3 nebo novější, zvažte nastavení parametru ladění pro použití plného výkonu pro přechodné zprávy.

Související úlohy

“Zapnutí a vypnutí dopředného čtení” na stránce 249
Při výchozím nastavení je dopředné čtení vypnuto. Čtení napřed můžete povolit na úrovni fronty nebo aplikace.

Související odkazy

“Volby MQGET a dopředné čtení” na stránce 248
Je-li povoleno čtení napřed, nejsou podporovány všechny volby MQGET; některé volby jsou vyžadovány k konzistenci mezi voláními MQGET.

Volby MQGET a dopředné čtení

Je-li povoleno čtení napřed, nejsou podporovány všechny volby MQGET; některé volby jsou vyžadovány k konzistenci mezi voláními MQGET.

Při volání MQOPEN s MQOH_READAHEAD klient WebSphere MQ povolí čtení napřed, pokud jsou splněny určité podmínky. Tyto podmínky zahrnují:

- Jak klient, tak i vzdálený správce front musí používat produkt WebSphere MQ verze 7 nebo novější.
- Klientská aplikace musí být zkompilována a propojena s knihovnami klienta WebSphere MQ MQI s podporou podprocesů.
- Kanál klienta musí používat protokol TCP/IP.
- Kanál musí mít nenulové nastavení SharingConversations (SHARECNV) v definici kanálu klienta i serveru.

Následující tabulka uvádí, které volby jsou podporovány pro použití s dopředným čtením a zda mohou být změněny mezi voláními MQGET.

Tabulka 40. Volby MQGET a dopředné čtení			
	Povoleno, je-li dopředné čtení povoleno a lze je měnit mezi voláními MQGET[5]	**Povoleno, je-li dopředné čtení povoleno, ale nelze je měnit mezi voláními MQGET[1]**	**Volby MQGET, které nejsou povoleny, je-li povoleno čtení napřed[2]**
MQGET MQMD	MsgId[3] CorrelId[3]	Kódování CodedCharSetId	
Volby MQGMO MQGET	• MQGMO_NO_WAIT • MQGMO_BROWSE_MESSAGE _UNDER_CURSOR • NEJPRVE MQGMO_BROWSE_FIRST • PŘÍŠTĚ MQGMO_BROWSE_NEXT • FUNKCE MQGMO_FAIL_IF_QUIESCING	• MQGMO_SYNCPOINT_IF _PERSISTENT • MQGMO_NO_SYNCPOINT • MQGMO_ACCEPT_ ZKRÁCENÁ _ZPRÁVA • MQGMO_CONVERT	• SIGNÁL MQGMO_SET_DATA • MQGMO_SYNCPOINT • MQGMO_MARKER_SKIP _BACKOUT • MQGMO_MSG_UNDER _CURSOR[4] • MQGMOVÝ_ZÁMEK • MQGMO_ODEMKNOUT • MQGMO_LOGICAL_ORDER • ZPRÁVA MQGMO_COMPLETE_MSG • MQGMO_ALL_MSGS_AVAILABLE • MQGMO_ALL_SEGMENTS_ K DISPOZICI

Notes:

1. Pokud jsou tyto volby upraveny mezi voláními MQGET, je vrácen kód příčiny MQRC_OPTIONS_CHANGED.
2. Pokud se tyto volby zadaly při prvním volání MQGET, bude dopředné čtení zablokováno. Budou-li tyto volby zadány při následném volání MQGET, vrátí se kód příčiny MQRC_OPTIONS_ERROR.
3. Pokud aplikace klienta změní hodnoty MsgId a CorrelId mezi voláními MQGET, zprávy s předchozími hodnotami již mohly být odeslány klientovi a zůstanou ve vyrovnávací paměti pro čtení napřed klienta, dokud nebudou spotřebovány (nebo automaticky vymazány).
4. MQGMO_MSG_UNDER_CURSOR nelze použít s dopředným čtením. Čtení napřed je zakázáno, když jsou při otevření fronty určeny volby MQOO_BROWSE a jeden z voleb MQOO_INPUT_SHARED nebo MQOO_INPUT_EXCLUSIVE.
5. Je-li dopředné čtení povoleno, první příkaz MQGET určuje, zda mají být zprávy procházeny nebo načíteny z fronty. Pokud klientská aplikace poté použije příkaz MQGET se změněnými volbami, jako například při pokusu o procházení po počátečním získání nebo při pokusu o získání následujícího počátečního procházení, je vrácen kód příčiny MQRC_OPTIONS_CHANGED.

Pokud klient změní svá kritéria výběru mezi voláními MQGET, zprávy uložené v vyrovnávací paměti dopředného čtení, které odpovídají počátečnímu kritériu výběru, nejsou aplikacemi klienta spotřebovávány a zůstávají uvízlé v paměti dopředného čtení klienta. V situacích, kdy vyrovnávací paměť pro čtení napřed klienta obsahuje mnoho uvízlých zpráv, výhody spojené s dopředným čtením se ztratí a pro každou spotřebovanou zprávu se vyžaduje zvláštní požadavek na server. Chcete-li určit, zda je čtení napřed používáno efektivně, můžete použít parametr stavu připojení, READA.

Čtení napřed může být zablokované, je-li to požadováno aplikací kvůli nekompatibilním volbám uvedeným na prvním volání MQGET. V této situaci stav připojení ukazuje dopředné čtení jako zablokované.

Pokud se kvůli těmto omezením na MQGET rozhodnete, že návrh aplikace klienta není vhodný pro čtení napřed, určete volbu MQOPEN MQOO_READ_AHEAD_NO. Nebo nastavte výchozí hodnotu dopředného čtení u fronty, která se otevírá, změněna na hodnotu NO nebo DISABLED.

Zapnutí a vypnutí dopředného čtení

Při výchozím nastavení je dopředné čtení vypnuto. Čtení napřed můžete povolit na úrovni fronty nebo aplikace.

Informace o této úloze

Při volání MQOPEN s MQOH_READAHEAD klient WebSphere MQ povolí čtení napřed, pokud jsou splněny určité podmínky. Tyto podmínky zahrnují:

- Jak klient, tak i vzdálený správce front musí používat produkt WebSphere MQ verze 7 nebo novější.
- Klientská aplikace musí být zkompilována a propojena s knihovnami klienta WebSphere MQ MQI s podporou podprocesů.

- Kanál klienta musí používat protokol TCP/IP.
- Kanál musí mít nenulové nastavení SharingConversations (SHARECNV) v definici kanálu klienta i serveru.

Povolení dopředného čtení:

- Chcete-li konfigurovat dopředné čtení na úrovni fronty, nastavte atribut queue, DEFREADA na YES.
- Chcete-li konfigurovat čtení napřed na úrovni aplikace, postupujte takto:
  - chcete-li při volání funkce MQOPEN používat funkci MQOO_READ_AHEAD, je-li to možné, použijte volbu dopředného čtení. Pokud byl atribut fronty DEFREADA nastaven na hodnotu DISABLED, není možné, aby aplikace klienta používala čtení napřed.
  - chcete-li používat dopředné čtení pouze v případě, že je ve frontě povoleno čtení napřed, použijte volbu MQOO_READ_AHEAD_AS_Q_DEF u volání funkce MQOPEN.

Není-li návrh klientské aplikace vhodný pro čtení napřed, můžete jej zakázat:

- na úrovni fronty nastavením atributu fronty DEFREADA na NO, pokud nechcete použít dopředné čtení, pokud není vyžádáno klientskou aplikací, nebo VYPNUTÉ, pokud nechcete, aby bylo čtení napřed používáno bez ohledu na to, zda je dopředné čtení vyžadováno klientskou aplikací.
- na úrovni aplikace pomocí volby MQOO_NO_READ_AHEAD při volání funkce MQOPEN.

Dvě volby MQCLOSE vám umožňují konfigurovat, co se stane se všemi zprávami, které jsou ukládány do vyrovnávací paměti dopředného čtení, pokud je fronta uzavřena.

- Použijte MQCO_IMMEDIATE k zahození zpráv do vyrovnávací paměti dopředného čtení.
- Použijte MQCO_QUIESCE, abyste zajistili, že zprávy ve vyrovnávací paměti dopředného čtení budou spotřebovány aplikací před zavřením fronty. Je-li zadán příkaz MQCLOSE s hodnotou MQCO_QUIESCE a existují zprávy, které zbývají do vyrovnávací paměti čtení napřed, funkce MQRC_READ_AHEAD_MSGS se vrátí s hodnotou MQCC_WARNING.

Vyladění výkonu pro přechodné zprávy v produktu AIX

Pokud používáte produkt AIX V5.3 nebo novější, zvažte nastavení parametru ladění pro použití plného výkonu pro přechodné zprávy.

Chcete-li nastavit parametr ladění tak, aby byl účinný okamžitě, zadejte jako uživatel root následující příkaz:

```
/usr/sbin/ioo -o j2_nPagesPerWriteBehindCluster=0
```

Chcete-li nastavit parametr ladění tak, aby se okamžitě nabyl účinku a přetrvá po opětovném zavedení systému, zadejte jako uživatel root následující příkaz:

```
/usr/sbin/ioo -p -o j2_nPagesPerWriteBehindCluster=0
```

Normálně se přechodné zprávy uchovávají pouze v paměti, ale existují okolnosti, kdy produkt AIX může naplánovat zápis přechodných zpráv na disk. Zprávy, které mají být zapsány na disk, jsou nedostupné pro příkaz MQGET až do dokončení zápisu na disk. Navrhovaný příkaz vyladění tuto prahovou hodnotu liší; místo plánování zápisu zpráv na disk, je-li ve frontě uvedeno 16 kilobajtů dat, dochází k zápisu na disk pouze v případě, že se skutečné úložiště na počítači blíží plné. To je globální změna a může ovlivnit další softwarové komponenty.

On AIX, when using multithreaded applications and especially when running on machines with multiple processors, we strongly recommend setting AIXTHREAD_SCOPE=S in the mqm id `.profile` or setting AIXTHREAD_SCOPE=S in the environment before starting the application, for better performance and more solid scheduling. Příklad:

```
export AIXTHREAD_SCOPE=S
```

Nastavení AIXTHREAD_SCOPE=S znamená, že uživatelské podprocesy vytvořené s výchozími atributy jsou umístěny do rozsahu soupeření celého systému. Je-li uživatelský podproces vytvořen s rozsahem soupeření celého systému, je svázán s vláknem jádra a je naplánován jádrem. Základní podproces jádra není sdílen s žádným jiným uživatelským podprocesem.

Deskriptory souborů

Při spuštění vícevláknového procesu, jako je proces agenta, můžete dosáhnout měkkého limitu pro deskriptory souborů. Tento limit vám poskytne kód příčiny IBM WebSphere MQ `MQRC_UNEXPECTED_ERROR (2195)` a v případě, že existuje dostatek deskriptorů souborů, soubor IBM WebSphere MQ FFST™ .

Chcete-li se tomuto problému vyhnout, můžete zvýšit limit procesu pro počet deskriptorů souboru. Chcete-li to provést, změňte atribut `nofiles` v `/etc/security/limits` na 10.000 pro ID uživatele mqm nebo ve výchozí stanze.

Limity systémových prostředků

Nastavte omezení systémových prostředků pro segment dat a segment zásobníku na neomezený počet pomocí následujících příkazů v příkazovém řádku:

```
ulimit -d unlimited
ulimit -s unlimited
```

Zpracování zpráv větších než 4 MB

Zprávy mohou být příliš velké pro aplikaci, frontu nebo správce front. V závislosti na daném prostředí nabízí produkt WebSphere MQ řadu způsobů práce se zprávami, které jsou delší než 4 MB.

Můžete zvýšit atribut *MaxMsgLength* až na 100 MB ve všech systémech WebSphere MQ na adrese V6 nebo pozdější. Nastavte tuto hodnotu tak, aby odrážela velikost zpráv používajících frontu. V systémech WebSphere MQ jiných než WebSphere MQ for z/OSmůžete také:

1. Použijte segmentované zprávy. (Zprávy mohou být segmentovány buď aplikací, nebo správcem front.)
2. Použít referenční zprávy.

Každý z těchto přístupů je popsán ve zbývající části této části.

Zvýšení maximální délky zprávy

Atribut správce front *MaxMsgLength* definuje maximální délku zprávy, kterou může správce front zpracovat. Podobně atribut fronty *MaxMsgLength* je maximální délka zprávy, kterou lze zpracovat pomocí fronty. Podporovaná maximální délka zprávy *výchozí* závisí na prostředí, ve kterém pracujete.

Pokud obsluhujete velké zprávy, můžete tyto atributy změnit nezávisle na sobě. Hodnotu atributu správce front můžete nastavit v rozsahu 32768 bajtů až 100 MB; můžete nastavit hodnotu atributu fronty v rozsahu 0 až 100 MB.

Poté, co změníte jeden nebo oba atributy *MaxMsgLength* , restartujte aplikace a kanály, abyste se ujistili, že se změny projeví.

Po provedení těchto změn musí být délka zprávy menší nebo rovna než atributy fronty a správce front *MaxMsgLength* . Existující zprávy však mohou být delší než jeden z těchto atributů.

Je-li zpráva pro frontu příliš velká, vrátí se hodnota MQRC_MSG_TOO_BIG_FOR_Q. Podobně, je-li zpráva pro správce front příliš velká, vrátí se hodnota MQRC_MSG_TOO_BIG_FOR_Q_MGR.

Tato metoda manipulace s velkými zprávami je snadná a pohodlná. Před použitím však zvažte následující faktory:

- Jednotnost správců front se snižuje. Maximální velikost dat zprávy je určena serverem *MaxMsgLength* pro každou frontu (včetně přenosových front), na které bude zpráva vložena. Tato hodnota je často

standardně nastavena na hodnotu *MaxMsgLength*správce front, zvláště pro přenosové fronty. Proto je obtížné předpovídat, zda je zpráva příliš velká, když má cestovat do vzdáleného správce front.

- Využití systémových prostředků se zvýšilo. Aplikace například potřebují větší vyrovnávací paměti a na některých platformách může dojít k vyššímu využití sdíleného úložiště. Úložný prostor fronty by měl být ovlivněn pouze tehdy, je-li to požadováno pro větší zprávy.
- Postižení kanálu kanálu je ovlivněno. Velká zpráva je stále započítává pouze jako jedna zpráva k počtu dávek, ale potřebuje delší dobu přenosu, čímž se zvyšuje doba odezvy pro ostatní zprávy.

Segmentace zpráv

Tyto informace použijte k seznámení se s segmentováním zpráv.

Poznámka: Nepodporováno v produktu IBM WebSphere MQ for z/OS nebo aplikacemi, které používají třídy IBM WebSphere MQ pro platformu JMS.

Zvýšení maximální délky zprávy, jak je vysvětleno v tématu “Zvýšení maximální délky zprávy” na stránce 251 , má určité negativní důsledky. Tato zpráva může také způsobit, že zpráva je příliš velká pro frontu nebo správce front. V těchto případech můžete segmentovat zprávu. Další informace o segmentech viz “Skupiny zpráv” na stránce 34.

Další sekce se podívejte na běžné použití pro segmentování zpráv. Při získávání a destruktivním získávání se předpokládá, že volání MQPUT nebo MQGET *vždy* fungují v rámci pracovní jednotky. Vždy zvažte použití této techniky, aby se snížila možnost přítomnosti neúplných skupin v síti. Předpokládá se jednofázové potvrzení správce front, ale jiné metody koordinace jsou stejně platné.

Kromě toho se při získávání aplikací předpokládá, že pokud více serverů zpracovává stejnou frontu, každý server zpracuje podobný kód, takže jeden server nikdy neselže při hledání zprávy nebo segmentu, které očekává, že tam bude (protože bylo zadáno MQGMO_ALL_MSGS_AVAILABLE nebo MQGMO_ALL_SEGMENTS_AVAILABLE).

Vložení a získání segmentované zprávy, která obsahuje jednotky práce

Můžete vložit a získat segmentovanou zprávu obsahující jednotku práce podobným způsobem, jako je “Uvedení a získání skupiny, která zahrnuje jednotky práce” na stránce 243.

Nemůžete však vložit nebo načíst segmentované zprávy v rámci globální transakce.

Segmentace a opětovné sestavení pro správce front

Jedná se o nejjednodušší scénář, ve kterém jedna aplikace vkládá zprávu, která má být načtena jinou aplikací. Zpráva může být velká: není příliš velká pro vložení nebo získání aplikace na zpracování v jedné vyrovnávací paměti, ale příliš velká pro správce front nebo frontu, na kterou má být zpráva vložena.

Jediné změny, které jsou nezbytné pro tyto aplikace, jsou určeny k tomu, aby aplikace mohla autorizovat správce front, aby provedl segmentaci, je-li to nutné:

```
PMO.Options = (existing options)
MD.MsgFlags = MQMF_SEGMENTATION_ALLOWED
memcpy(MD.GroupId, MQGI_NONE, MQ_GROUP_ID_LENGTH)
MQPUT
```

a v případě aplikace, která má požádat správce front o opětné sestavení zprávy, pokud byla segmentována:

```
GMO.Options = MQGMO_COMPLETE_MSG | (existing options)
MQGET
```

V tomto nejjednodušším scénáři musí aplikace před voláním MQPUT resetovat pole GroupId na hodnotu MQGI_NONE, aby správce front mohl generovat jedinečný identifikátor skupiny pro každou zprávu. Pokud to není provedeno, nespřízněné zprávy mohou mít stejný identifikátor skupiny, což může následně vést k nesprávnému zpracování.

Vyrovnávací paměť aplikace musí být dostatečně velká, aby mohla obsahovat znovu sestavovanou zprávu (pokud nezahrnete volbu MQGMO_ACCEPT_TRUNCATED_MSG).

Pokud má být atribut MAXMSGLEN fronty upraven tak, aby pojmul segmentaci zpráv, pak zvažte:

- Minimální segment zpráv podporovaný v lokální frontě je 16 bajtů.
- Pro přenosovou frontu musí parametr MAXMSGLEN také obsahovat prostor potřebný pro záhlaví. Zvažte použití hodnoty o velikosti alespoň 4000 bajtů větší než maximální očekávaná délka uživatelských dat v libovolném segmentu zprávy, který lze vložit do přenosové fronty.

Je-li konverze dat nezbytná, může ji aplikace vyžadovat zadáním hodnoty MQGMO_CONVERT. To by mělo být jednoduché, protože uživatelská procedura pro převod dat se zobrazí spolu s úplnou zprávou. Nepokoušejte se převést data v odesílacím kanálu, je-li zpráva segmentovaná, a formát dat je takový, že uživatelská procedura konverze dat nemůže provést převod na neúplných datech.

Segmentace aplikace
Segmentace aplikace se používá, když segmentace správce front není vhodná, nebo když aplikace vyžadují konverzi dat se specifickými hranicemi segmentu.

Segmentace aplikace se používá ze dvou hlavních důvodů:

1. Samotná segmentace správce front není adekvátní, protože je zpráva příliš velká, aby ji bylo možné zpracovat v rámci jedné vyrovnávací paměti aplikacemi.
2. Převod dat musí být proveden odesílacími kanály a formát je takový, že aplikace musí stanovit, kde mají být hranice segmentu, aby mohly být konverze jednotlivých segmentů možné.

Pokud však převod dat není problém, nebo pokud aplikace získání vždy používá MQGMO_COMPLETE_MSG, segmentaci správce front lze také povolit zadáním hodnoty MQMF_SEGMENTATION_ALLOWED. V našem příkladu aplikace segmentuje zprávu do čtyř segmentů:

```
PMO.Options = MQPMO_LOGICAL_ORDER | MQPMO_SYNCPOINT

MQPUT MD.MsgFlags = MQMF_SEGMENT
MQPUT MD.MsgFlags = MQMF_SEGMENT
MQPUT MD.MsgFlags = MQMF_SEGMENT
MQPUT MD.MsgFlags = MQMF_LAST_SEGMENT

MQCMIT
```

Pokud nepoužíváte MQPMO_LOGICAL_ORDER, aplikace musí nastavit *Offset* a délku každého segmentu. V tomto případě se logický stav neudržuje automaticky.

Aplikace pro získání nemůže zaručit, že bude mít vyrovnávací paměť dostatečně velkou vyrovnávací paměť, aby mohla být zadržena znovu sestavená zpráva. Musí být proto připraven zpracovávat jednotlivé segmenty jednotlivě.

U zpráv, které jsou segmentovány, nechce tato aplikace začít zpracovávat jeden segment, dokud nejsou přítomny všechny segmenty tvořící logickou zprávu. Funkce MQGMO_ALL_SEGMENTS_AVAILABLE je proto určena pro první segment. Zadáte-li MQGMO_LOGICAL_ORDER a aktuální logická zpráva, bude hodnota MQGMO_ALL_SEGMENTS_AVAILABLE ignorována.

Po načtení prvního segmentu logické zprávy použijte MQGMO_LOGICAL_ORDER, abyste se ujistili, že zbývající segmenty logické zprávy jsou načteny v pořadí.

Ve zprávách v různých skupinách není věnována žádná pozornost. Pokud se takové zprávy vyskytnou, jsou zpracovány v pořadí, ve kterém se první segment každé zprávy vyskytuje ve frontě.

```
GMO.Options = MQGMO_SYNCPOINT | MQGMO_LOGICAL_ORDER
            | MQGMO_ALL_SEGMENTS_AVAILABLE | MQGMO_WAIT
do while ( SegmentStatus == MQSS_SEGMENT )
   MQGET
   /* Process each remaining segment of the logical message */
   ...

MQCMIT
```

Segmentace aplikace logických zpráv
Zprávy musí být udržovány v logickém pořadí ve skupině a některé nebo všechny z nich by mohly být tak velké, že vyžadují segmentaci aplikace.

V našem příkladu má být vložena skupina čtyř logických zpráv. Všechny kromě třetí zprávy jsou velké a vyžadují segmentaci, kterou provádí aplikace vkládání:

```
PMO.Options = MQPMO_LOGICAL_ORDER | MQPMO_SYNCPOINT

MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP      | MQMF_SEGMENT
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP      | MQMF_SEGMENT
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP      | MQMF_LAST_SEGMENT

MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP      | MQMF_SEGMENT
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP      | MQMF_LAST_SEGMENT

MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP

MQPUT MD.MsgFlags = MQMF_LAST_MSG_IN_GROUP | MQMF_SEGMENT
MQPUT MD.MsgFlags = MQMF_LAST_MSG_IN_GROUP | MQMF_SEGMENT
MQPUT MD.MsgFlags = MQMF_LAST_MSG_IN_GROUP | MQMF_LAST_SEGMENT

MQCMIT
```

V aplikaci GET je MQGMO_ALL_MSGS_AVAILABLE uveden v první MQGET. To znamená, že se nenačtou žádné zprávy nebo segmenty skupiny, dokud nebude k dispozici celá skupina. Když byla načtena první fyzická zpráva skupiny, je použit MQGMO_LOGICAL_ORDER, aby se zajistilo, že segmenty a zprávy skupiny se načtou v pořadí:

```
GMO.Options = MQGMO_SYNCPOINT | MQGMO_LOGICAL_ORDER
            | MQGMO_ALL_MSGS_AVAILABLE | MQGMO_WAIT

do while ( (GroupStatus   != MQGS_LAST_MSG_IN_GROUP) ||
             (SegmentStatus != MQGS_LAST_SEGMENT) )
   MQGET
   /* Process a segment or complete logical message.  Use the GroupStatus
      and SegmentStatus information to see what has been returned */
   ...

MQCMIT
```

Poznámka: Pokud uvedete MQGMO_LOGICAL_ORDER a existuje aktuální skupina, MQGMO_ALL_MSGS_AVAILABLE se ignoruje.

Referenční zprávy
Tyto informace použijte k získání více informací o referenčních zprávách.

Poznámka: Nepodporováno v produktu WebSphere MQ pro z/OS.

Tato metoda umožňuje přenos velkého objektu z jednoho uzlu do jiného bez uložení objektu ve frontách produktu WebSphere MQ ve zdrojovém nebo v cílovém uzlu. To je zvláště výhodné v případě, že data existují v jiné formě, například pro poštovní aplikace.

Chcete-li tak učinit, zadejte uživatelskou proceduru pro zprávy na obou koncích kanálu. Další informace o tom, jak to provést, viz "Ukončovací programy zpráv kanálu" na stránce 395.

WebSphere MQ definuje formát záhlaví referenční zprávy (MQRMH). Popis naleznete v tématu MQRMH . Tato hodnota je rozpoznána s definovaným názvem formátu a může být následována skutečnými daty.

Chcete-li zahájit přenos velkého objektu, aplikace může vložit zprávu obsahující záhlaví referenční zprávy bez dat, která následují. Když tato zpráva opustí uzel, výstupní program zprávy načte příslušný objekt vhodným způsobem a připojí jej k referenční zprávě. Pak vrátí zprávu (nyní větší než dříve) odesílajícímu agentovi MCA pro přenos do přijímajícího agenta MCA.

Další uživatelská procedura pro zprávy je konfigurována v přijímajícím agentu MCA. Když tento výstup zprávy přijme jednu z těchto zpráv, vytvoří objekt s pomocí dat objektu, která byla přidána, a předá na referenční zprávu *bez* . Referenční zpráva může být nyní přijata aplikací a tato aplikace ví, že objekt (nebo alespoň část zastupovaného touto referenční zprávou) byl vytvořen v tomto uzlu.

Maximální množství dat objektů, které může odesílající uživatelská procedura zprávy připojit k referenční zprávě, je omezena vyjednanou maximální délkou zprávy pro kanál. Ukončení může vrátit pouze jednu zprávu pro práci s agentem MCA pro každou zprávu, že je předána, takže aplikace umisťování může vložit několik zpráv, aby mohl být jeden objekt přenesen. Každá zpráva musí identifikovat *logickou* délku a posun objektu, který má být k němu připojen. Avšak v případech, kdy není možné zjistit celkovou velikost objektu nebo maximální velikost povolenou kanálem, navrhněte odesílající uživatelskou proceduru tak, aby vkládaná aplikace vkládala pouze jednu zprávu a uživatelská procedura umístí další zprávu do přenosové fronty, jakmile ji přidá tolik dat, kolik může být posláno ke zprávě.

Před použitím této metody práce s velkými zprávami zvažte následující body:

- Agent MCA a uživatelská procedura pro zprávy jsou spuštěny pod ID uživatele produktu WebSphere MQ . Uživatelská procedura zprávy (a proto ID uživatele) potřebuje mít přístup k objektu, aby jej buď získal na odesílající straně, nebo jej vytvořil na přijímajícím konci; to může být proveditelné pouze v případech, kdy je objekt univerzálně přístupný. Vzniká tak otázka zabezpečení.
- Pokud se referenční zpráva s hromadně připojenými daty musí projít několika správci front před dosažením cíle, jsou hromadná data *jsou* přítomna ve frontách produktu WebSphere MQ na uzlech, které zasahují do těchto uzlů. V těchto případech však není třeba žádná zvláštní podpora nebo východy.
- Návrh ukončení zprávy bude ztížit, je-li povoleno přesměrování nebo řazení do fronty s dead-letter. V těchto případech může dojít k tomu, že se části objektu dostanou mimo pořadí.
- Když na místo určení dorazí referenční zpráva, přijímající uživatelská procedura zprávy vytvoří objekt. Avšak, toto není synchronizováno s pracovní jednotkou MCA, takže pokud je dávka vrácena, další referenční zpráva obsahující tuto stejnou část objektu bude doručena do pozdější dávky a uživatelská procedura zprávy se může pokusit znovu vytvořit stejnou část objektu. Je-li objekt například řada aktualizací databáze, může být tato situace nepřijatelná. Je-li tomu tak, uživatelská procedura pro zprávy musí uchovávat protokol, v němž byly provedeny aktualizace; to může vyžadovat použití fronty produktu WebSphere MQ .
- V závislosti na charakteristice typu objektu může dojít k tomu, že zpráva se ukončí a aplikace budou muset spolupracovat při zachování počtu použití, aby mohl být objekt vymazán, když již není potřeba. Identifikátor instance může být také povinný; v záhlaví zprávy odkazu je k dispozici pole (viz MQRMH).
- Je-li jako distribuční seznam vložena referenční zpráva, musí být objekt vyhledatelný pro každý výsledný distribuční seznam nebo cílové místo určení v daném uzlu. Je možné, že budete potřebovat zachovat počty použití. Zvažte také možnost, že by uzel mohl být konečným uzlem pro některá místa určení v seznamu, ale zprostředkující uzel pro ostatní.
- Hromadná data se obvykle nekonvertují. Důvodem je to, že ke konverzi dochází *před* vyvoláním uživatelské procedury pro zpracování zprávy. Z tohoto důvodu nesmí být konverze požadována na původním odesílacím kanálu. Pokud se referenční zpráva předává přes přechodný uzel, hromadná data se konvertují při odeslání z mezilehlého uzlu, je-li požadováno.
- Referenční zprávy nemohou být segmentovány.

Použití struktur MQRMH a MQMD

Viz MQRMH a MQMD pro popis polí v záhlaví referenční zprávy a v deskriptoru zpráv.

Ve struktuře MQMD nastavte pole *Format* na hodnotu MQFMT_REF_MSG_HEADER. Formát MQHREF je při požadavku na příkaz MQGET automaticky převeden produktem WebSphere MQ spolu s veškerými hromadným daty, která jsou následující.

Dále je uveden příklad použití polí *DataLogicalOffset* a *DataLogicalLength* MQRMH:

Aplikace uvedení do provozu může obsahovat referenční zprávu obsahující:

- Žádná fyzická data
- *DataLogicalLength* = 0 (tato zpráva představuje celý objekt)
- *DataLogicalOffset* = 0.

Za předpokladu, že objekt má délku 70 000 bajtů, odesílající uživatelská procedura odešle prvních 40 000 bajtů v kanálu v referenční zprávě obsahující:

- 40 000 bajtů fyzických dat za MQRMH
- *DataLogicalLength* = 40000
- *DataLogicalOffset* = 0 (od začátku objektu).

Poté umístí jinou zprávu do přenosové fronty obsahující:

- Žádná fyzická data
- *DataLogicalLength* = 0 (až do konce objektu). Můžete zde zadat hodnotu 30 000.
- *DataLogicalOffset* = 40000 (od tohoto bodu).

Je-li tato uživatelská procedura pro odeslání zprávy zobrazena uživatelskou procedurou odeslání zprávy, je připojeno zbývajících 30 000 bajtů dat a pole jsou nastavena na následující hodnoty:

- 30 000 bajtů fyzických dat za MQRMH
- *DataLogicalLength* = 30000
- *DataLogicalOffset* = 40000 (od tohoto bodu).

Příznak MQRMHF_LAST je také nastaven.

Popis ukázkových programů poskytovaných pro použití referenčních zpráv naleznete v tématu “Ukázkové programy pro distribuované platformy” na stránce 93.

Čekání na zprávy

Pokud chcete, aby program čekal na příchod zprávy do fronty, zadejte volbu MQGMO_WAIT v poli *Options* struktury MQGMO.

Použijte pole *WaitInterval* struktury MQGMO k určenímaximální doba (v milisekundách), po kterou má volání MQGET čekat na příchod zprávy do fronty.

Pokud se zpráva nedostaví do této doby, volání MQGET se dokončí s kódem příčiny MQRC_NO_MSG_AVAILABLE.

Můžete zadat neomezený interval čekání pomocí konstanty MQWI_UNLIMITED v poli *WaitInterval* . Události mimo váš ovládací prvek však mohou způsobit čekání vašeho programu na delší dobu, takže tuto konstantu použijte opatrně. Aplikace IMS nesmí určovat neomezený interval čekání, protože by se zabránilo ukončení systému IMS . (Je-li IMS ukončeno, vyžaduje ukončení všech závislých oblastí.) Místo toho mohou aplikace IMS určovat konečný interval čekání; pak, pokud je volání dokončeno bez načtení zprávy po tomto intervalu, vydejte další volání MQGET s volbou čekání.

Poznámka: Pokud více než jeden program čeká ve stejné sdílené frontě na *odebrání* zprávy, aktivuje se příchozí zpráva pouze jeden program. Avšak pokud čeká na prohlížení zprávy více než jeden program, všechny programy lze aktivovat. Další informace naleznete v popisu pole *Options* struktury MQGMO v produktu MQGMO.

Pokud se stav fronty nebo správce front změní před vypršením čekací doby, dojde k následujícím akcím:

- Pokud správce front přejde do klidového stavu a vy jste použili volbu MQGMO_FAIL_IF_QUIESCING, čekání je zrušeno a volání MQGET se dokončí s kódem příčiny MQRC_Q_MGR_QUIESCING. Bez této volby zůstává volání čekání.
- Je-li správce front nucen zastavit nebo je zrušen, je volání MQGET dokončeno buď s kódem příčiny MQRC_Q_MGR_STOPPING nebo MQRC_CONNECTION_BROKEN.
- Změní-li se atributy fronty (nebo fronty, na kterou se název fronty řeší), takže požadavky na získání jsou nyní blokovány, čekání je zrušeno a volání MQGET se dokončí s kódem příčiny MQRC_GET_INHIBITED.
- Změní-li se atributy fronty (nebo fronty, na kterou se název fronty řeší) takovým způsobem, že je vyžadována volba FORCE, bude čekání zrušeno a volání MQGET se dokončí s kódem příčiny MQRC_OBJECT_CHANGED.

Další informace o okolnostech, za kterých k těmto akcím dochází, najdete v tématu MQGMO.

Vynechání odvolání

Aplikační program můžete zabránit v zadání smyčky *MQGET-error-backout* zadáním volby **MQGMO_MARK_SKIP_BACKOUT** na volání MQGET.

Poznámka: Podporováno pouze v produktu WebSphere MQ for z/OS.

Jako součást pracovní jednotky může aplikační program vydat jeden nebo více volání MQGET k získání zpráv z fronty. Pokud aplikační program zjistí chybu, může vrátit jednotku práce zpět. Tím se obnoví všechny prostředky aktualizované během této jednotky práce do stavu, ve kterém se nacházely před spuštěním jednotky práce, a znovu obnoví zprávy načtené voláním MQGET.

Po opětovném zavedení jsou tyto zprávy k dispozici pro následující volání MQGET vydaná aplikačním programem. V mnoha případech to nezpůsobí problém pro aplikační program. Avšak v případech, kdy nelze obcházet chybu vedoucí k vrácení zpět, může mít zpráva ve frontě znovu zavedena zpráva, která může způsobit, že aplikační program vstoupí do smyčky *MQGET-error-backout* .

Chcete-li tomuto problému předejít, zadejte v rámci příkazu MQGET volbu MQGMO_MARK_Send_KIP_BACOUT. Tím se označí požadavek MQGET jako nezahrnutý do odvolání zahájeného aplikací; to znamená, že nesmí být vrácena. Použití této volby znamená, že když dojde k odvolání, aktualizace z jiných prostředků se zálohují podle potřeby, ale s označenou zprávou se zachází, jako by byla načtena pod novou pracovní jednotkou.

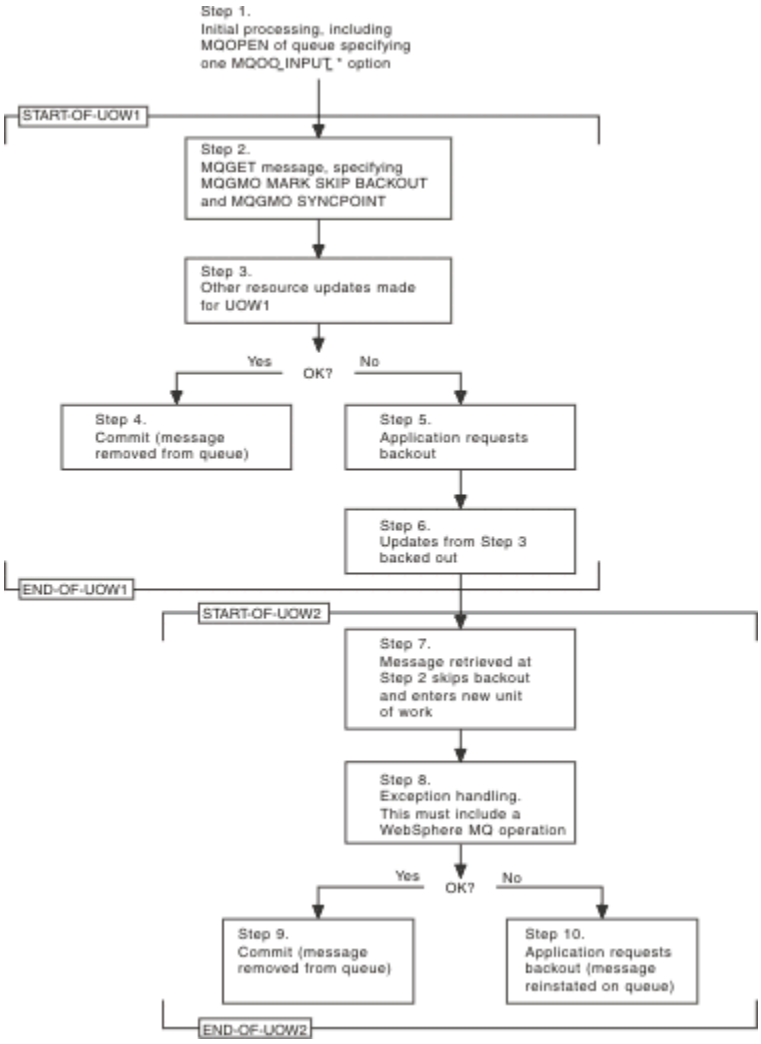
Aplikační program musí zadat volání produktu WebSphere MQ buď k potvrzení nové transakce, nebo k vrácení nové pracovní jednotky. Program může například provádět zpracování výjimek, jako je například informování odesílatele o tom, že zpráva byla zahozena, a potvrdit jednotku práce tak, aby byla odstraněna zpráva z fronty, pokud je nová pracovní jednotka vrácena (z jakéhokoli důvodu) zpráva byla znovu zavedena do fronty.

V rámci pracovní jednotky může existovat pouze jeden požadavek MQGET označený jako přeskočení odvolání. Může však existovat několik dalších zpráv, které nejsou označeny jako přeskočení z odvolání. Jakmile je zpráva označena jako přeskočena, všechna další volání MQGET v rámci pracovní jednotky, která určuje MQGMO_MARK_SKIP_BACKUP, selžou s kódem příčiny MQRC_SECOND_MARK_NOT_ALLOWED.

Poznámka:

1. Označená zpráva se přeskočí pouze v případě, že je jednotka práce, která ji obsahuje, ukončena požadavkem aplikace na jeho vrácení. Je-li jednotka práce vrácena z jiných příčin, je zpráva vrácena do fronty stejným způsobem, jako by byla, pokud nebyla označena, aby přeskočila odvolání.
2. Přeskočení odvolání není podporováno v rámci uložených procedur produktu DB2 , které se podílí na jednotkách práce řízených RRS. Například volání MQGET s volbou MQGMO_MARK_KIP_BACKOUT selže s kódem příčiny MQRC_OPTION_ENVIRONMENT_ERROR.

Obrázek 36 na stránce 258 znázorňuje typickou posloupnost kroků, které může aplikační program obsahovat, je-li požadován požadavek MQGET pro přeskočení odvolání.



Obrázek 36. Vynechání odvolání pomocí funkce MQGMO_MARK_SKIP_BACKUP

Kroky v produktu Obrázek 36 na stránce 258 jsou:

Krok 1
K počátečnímu zpracování dojde v rámci transakce, včetně volání MQOPEN pro otevření fronty (zadáním jedné z voleb MQOO_INPUT_ *, aby bylo možné získat zprávy z fronty v kroku 2).

Krok 2
Je volána funkce MQGET s hodnotami MQGMO_SYNCPOINT a MQGMO_MARK_SKIP_BACKUP. Funkce MQGMO_SYNCPOINT je vyžadována, protože operace MQGET musí být v rámci pracovní jednotky pro funkci MQGMO_MARK_SKIP_BAC_BE platnou. V produktu Obrázek 36 na stránce 258 je tato jednotka práce odkazována jako UOW1.

Krok 3
Další aktualizace prostředků se provádějí v části UOW1. Ty mohou zahrnovat další volání MQGET (vydaná bez MQGMO_MARK_SKIP_BACKUUT).

Krok 4
Všechny aktualizace z kroků 2 a 3 jsou dokončeny podle potřeby. Aplikační program potvrdí aktualizace a UOW1 skončí. Zpráva načtená z kroku 2 bude odebrána z fronty.

Krok 5
Některé z aktualizací z kroků 2 a 3 nejsou dokončeny podle potřeby. Aplikační program požaduje, aby aktualizace provedené během těchto kroků byly zálohovány.

Krok 6
Aktualizace provedené v kroku 3 jsou vráceny zpět.

Krok 7
Požadavek MQGET, který byl proveden v kroku 2, přeskočí a stane se součástí nové pracovní jednotky, UOW2.

Krok 8
UOW2 provádí zpracování výjimek v reakci na objekt UOW1 , který má být zálohován. (Například volání MQPUT do jiné fronty, označující, že došlo k problému, který způsobil, že byl UOW1 zálohován.)

Krok 9
Krok 8 se dokončí podle potřeby, aplikační program potvrdí aktivitu a ukončí se UOW2 . Vzhledem k tomu, že požadavek MQGET je součástí UOW2 (viz krok 7), způsobí toto potvrzení, že zpráva bude odebrána z fronty.

Krok 10
Krok 8 není dokončen, jak je požadováno, a aplikační program zálohuje UOW2. Protože požadavek na získání zprávy je součástí UOW2 (viz krok 7), je také zálohován a znovu vrácen do fronty. Nyní je k dispozici pro další volání MQGET vydaná tímto nebo jiným aplikačním programem (ve stejné podobě jako jakákoli jiná zpráva ve frontě).

Převod dat aplikace

Je-li to nezbytné, MCAs převádí deskriptor zprávy a data záhlaví do požadované znakové sady a kódování. Převod může provést buď konec odkazu (tj. lokální agent MCA nebo vzdálená MCA), může tento spoj provést.

Když aplikace vkládá zprávy do fronty, lokální správce front přidá řídicí informace do deskriptorů zpráv, aby usnadnil řízení zpráv, když jsou zpracovány správci front a MCAs. V závislosti na daném prostředí jsou datová pole záhlaví zprávy vytvořena ve znakové sadě a v kódování lokálního systému.

Když přesouváte zprávy mezi systémy, někdy je třeba převést data aplikace do znakové sady a kódování, které je vyžadováno přijímajícím systémem. To lze provést buď z aplikačních programů na přijímajícím systému, nebo pomocí MCA na odesílajícím systému. Je-li konverze dat podporována na přijímajícím systému, použijte aplikační programy k převedení dat aplikace, spíše než v závislosti na konverzi, která se již vyskytla na odesílajícím systému.

Data aplikace jsou převedena v rámci aplikačního programu, když určujete volbu MQGMO_CONVERT v poli *Options* struktury MQGMO předané na volání MQGET a *všechny* následující podmínky jsou pravdivé:

- Pole *CodedCharSetId* nebo *Encoding* nastavená ve struktuře MQMD přidružené ke zprávě ve frontě se liší od polí *CodedCharSetId* nebo *Encoding* nastavených ve struktuře MQMD určené v rámci volání MQGET.
- Pole *Format* ve struktuře MQMD přidružené ke zprávě není MQFMT_NONE.
- Hodnota *BufferLength* zadaná v rámci volání MQGET není nulová.
- Délka dat zprávy není nula.
- Správce front podporuje převod mezi poli *CodedCharSetId* a *Encoding* zadanými ve strukturách MQMD přidruženému ke zprávě a voláním MQGET. Viz CodedCharSetId a Encoding , kde jsou uvedeny podrobnosti o identifikátorech kódované znakové sady a podporovaných kódování počítače.
- Správce front podporuje převod formátu zpráv. Je-li pole *Format* struktury MQMD přidružené ke zprávě jedním z vestavěných formátů, může správce front zprávu převést. Pokud *Format* není jedním z vestavěných formátů, musíte napsat uživatelskou proceduru pro převod dat, abyste tuto zprávu převedli.

Pokud má odesílající agent MCA převést data, zadejte klíčové slovo CONVERT (YES) na definici každého odesílatele nebo kanálu serveru, pro který je konverze vyžadována. Pokud dojde k selhání konverze dat, zpráva se odešle do fronty DLQ v odesílajícím správci front a pole *Feedback* struktury MQDLH označuje důvod. Pokud nelze zprávu vložit do fronty nedoručených zpráv, kanál se zavře a nepřevedená zpráva zůstane v přenosové frontě. Data conversion within applications rather than at sending MCAs eliminate this situation.

Jako pravidlo jsou data ve zprávě, která jsou popsána jako *znaková* data pomocí vestavěného formátu nebo uživatelské procedury pro převod dat, převedena z kódované znakové sady použité ve zprávě na požadovanou a *číselná* pole jsou převedena na požadované kódování.

Další podrobnosti o konvencích zpracování konverze použitých při převádění vestavěných formátů a na informace o psaní vlastních uživatelských procedur pro převod dat naleznete v tématu “Zápis uživatelských procedur pro převod dat” na stránce 399. Viz také Národní jazyky a Kódování počítače , kde najdete informace o tabulkách jazykové podpory a o podporovaných kódování počítače.

Převod znaků nového řádku EBCDIC

Potřebujete-li zajistit, aby data odesílaná z platformy EBCDIC do ASCII jedna byla identická s daty, která obdržíte znovu, musíte řídit konverzi znaků znaku EBCDIC nového řádku.

To lze provést pomocí přepínače závislého na platformě, který vynutí produkt WebSphere MQ pro použití nezměněných převodních tabulek, ale musíte si být vědomi nekonzistentního chování, které by mohlo vést k výsledku.

Problém vzniká, protože znak nového řádku EBCDIC se nepřevádí konzistentně přes platformy nebo konverzní tabulky. Výsledkem je, že jsou-li data zobrazena na platformě ASCII, může být formátování nesprávné. Tím by bylo obtížné například vzdáleně spravovat systém IBM i z platformy ASCII pomocí RUNMQSC.

Další informace o konverzi dat ve formátu EBCDIC do formátu ASCII najdete v tématu Převod dat .

Procházení zpráv ve frontě

Tyto informace použijte k vyhledání informací o procházení zpráv ve frontě pomocí volání MQGET.

Chcete-li použít volání MQGET k procházení zpráv ve frontě, postupujte takto:

1. Chcete-li otevřít frontu pro procházení určením volby MQOO_BROWSE, volejte MQOPEN.
2. Chcete-li procházet první zprávu ve frontě, zavolejte příkaz MQGET s volbou MQGMOPRE_FIRST. Chcete-li vyhledat požadovanou zprávu, opakovaně volejte MQGET s volbou MQGMO_BROTE_NEXT, abyste prošli mnoha zprávami.

   musí nastavíte pole *MsgId* a *CorrelId* struktury MQMD na hodnotu null po každém volání MQGET za účelem zobrazení všech zpráv.
3. Chcete-li zavřít frontu, volejte příkaz MQCLOSE.

Kurzor procházení

Když otevírejte frontu pro procházení (MQOPEN), volání založí kurzor procházení pro použití s voláními MQGET, které používají jednu z voleb procházení. Můžete si myslet na kurzor procházení jako na logický ukazatel, který je umístěn před první zprávou ve frontě.

Můžete mít více než jeden kurzor procházení (z jednoho programu) tak, že zadáte několik požadavků MQOPEN pro stejnou frontu.

Když voláte MQGET pro procházení, použijte jednu z následujících možností ve struktuře MQGMO:

NEJPRVE MQGMO_BROWSE_FIRST
Získá kopii první zprávy, která splňuje podmínky určené ve struktuře MQMD.

PŘÍŠTĚ MQGMO_BROWSE_NEXT
Získá kopii další zprávy, která splňuje podmínky určené ve struktuře MQMD.

MQGMO_BROWSE_MSG_UNDER_CURSOR
Získá kopii zprávy, na kterou se aktuálně odkazuje kurzor, tj. ten, který byl naposledy načten pomocí volby MQGMO_BROFIRST FIRST nebo MQGMO_BRONEXT.

Ve všech případech zůstává zpráva ve frontě.

Když otevřete frontu, kurzor procházení je umístěn logicky těsně před první zprávou ve frontě. To znamená, že pokud provedete volání MQGET okamžitě po volání MQOPEN, můžete pomocí volby MQGMO_BROWSE NEXT procházet první zprávu; nemusíte používat volbu MQGMO_BROTED FIRST.

Pořadí, ve kterém se zprávy kopírují z fronty, je určeno atributem *MsgDeliverySequence* fronty. (Další informace viz "Pořadí, ve kterém jsou zprávy načítány z fronty" na stránce 235.)

- "Fronty v posloupnosti FIFO (první dovnitř, první ven)" na stránce 261
- "Fronty v posloupnosti priorit" na stránce 261
- "Nepotvrzené zprávy" na stránce 261
- "Změnit na pořadí fronty" na stránce 261
- "Použití indexu fronty" na stránce 261

Fronty v posloupnosti FIFO (první dovnitř, první ven)

První zpráva ve frontě v tomto pořadí je zpráva, která byla na frontě nejdelší.

Chcete-li číst zprávy sekvenčně ve frontě, použijte příkaz MQGMOROWSE_NEXT. Zobrazí se zprávy vložené do fronty během procházení, protože fronta v této posloupnosti má zprávy umístěné na konci. Když kurzor rozpozná, že dosáhla konce fronty, kurzor procházení zůstane tam, kde je a vrátí se s MQRC_NO_MSG_AVAILABLE. Můžete ji pak buď nechat čekat na další zprávy, nebo ji resetovat na začátek fronty s voláním MQGMOPRADE_FIRST.

Fronty v posloupnosti priorit

První zpráva ve frontě v tomto pořadí je zpráva, která byla ve frontě nejdelší a která má nejvyšší prioritu v době, kdy bylo vydáno volání MQOPEN.

Chcete-li číst zprávy ve frontě, použijte příkaz MQGMO_BROE_NEXT.

Kurzor procházení ukazuje na další zprávu, která pracuje od priority první zprávy, která má být dokončena se zprávou na nejnižší prioritě. Během této doby prochází všechny zprávy, které byly do fronty vloženy, pokud mají prioritu stejnou nebo nižší, než je zpráva identifikovaná aktuálním kurzorem procházení.

Všechny zprávy zařazené do fronty s vyšší prioritou lze procházet pouze následujícími způsoby:

- Otevřením fronty znovu procházet, v tomto okamžiku je vytvořen nový kurzor procházení
- Použití volby MQGMOPRE_FIRST

Nepotvrzené zprávy

Nepotvrzená zpráva není nikdy viditelná pro procházení; kurzor procházení přeskočí přes něj.

Zprávy v rámci pracovní jednotky nelze procházet, dokud není potvrzena jednotka práce. Zprávy neměňte jejich umístění ve frontě, takže přeskočené, nepotvrzené zprávy nebudou vidět, i když jsou *jsou* potvrzeny, pokud nepoužijete volbu MQGMOPRE_FIRST a znovu pracují s frontou.

Změnit na pořadí fronty

Je-li pořadí doručení zprávy změněno z priority na FIFO, zatímco ve frontě jsou zprávy, pořadí zpráv, které jsou již ve frontě, se nezmění. Zprávy přidané do fronty později, mají výchozí prioritu fronty.

Použití indexu fronty

Při procházení indexované fronty obsahující pouze zprávy s jednou prioritou (buď perzistentní, nebo přechodná, nebo obě) používá správce front index k procházení při použití určitých forem procházení.

Poznámka: Podporováno pouze v produktu WebSphere MQ for z/OS.

Pokud indexovaná fronta obsahuje pouze zprávy s jednou prioritou, používají se některé z následujících forem procházení:

1. Pokud je fronta indexována podle MSGID, požadavky na procházení, které předají MSGID ve struktuře MQMD, se zpracují s použitím indexu k nalezení cílové zprávy.

2. Pokud je fronta indexována podle CORRELID, procházejí požadavky, které projdou CORRELID ve struktuře MQMD, za použití indexu k nalezení cílové zprávy.
3. Pokud je fronta indexována pomocí GROUPID, procházejí požadavky, které předáte GROUPID ve struktuře MQMD, zpracované s použitím indexu k nalezení cílové zprávy.

Pokud požadavek na procházení nepředává ID MSGID, CORRELID nebo GROUPID ve struktuře MQMD, fronta je indexována a je vrácena zpráva, musí být nalezena položka indexu pro tuto zprávu a informace v ní použité k aktualizaci kurzoru pro procházení. Použijete-li široký výběr hodnot indexu, nepřidává se do požadavku na procházení významné další zpracování.

Procházení zpráv, pokud je délka zprávy neznámá

Chcete-li procházet zprávu, když neznáte velikost zprávy a nechcete použít pole *MsgId*, *CorrelId*nebo *GroupId* k vyhledání zprávy, můžete použít volbu MQGMOROWS_MSG_UNDER_CURSOR:

1. Zadejte příkaz MQGET s:
   - buď volba MQGMO_BROWSE_FIRST nebo MQGMO_BRONEXT NEXT
   - Volba MQGMO_ACCEPT_TRUNCATED_MSG
   - Nula vyrovnávací paměti

   Poznámka: Pokud je pravděpodobné, že jiný program získá stejnou zprávu, zvažte použití volby MQGMO_LOCK. Je třeba vrátit MQRC_TRUNCATED_MSG_ACCEPTED.
2. Chcete-li přidělit potřebnou paměť, použijte navrácenou hodnotu *DataLength* .
3. Zadejte příkaz MQGET s parametrem MQGMO_BROWS_MSG_UNDER_CURSOR.

Zpráva ukazovala na načtenou poslední, která byla načtena; kurzor procházení se nepřesunul. Můžete zvolit buď zamknutí zprávy pomocí volby MQGMO_LOCK, nebo odemknout zamčenou zprávu pomocí volby MQGMO_UNLOCK.

Volání selže, pokud nebyla úspěšně vydána žádná operace MQGET s volbami MQGMO_BROWSE_FIRST nebo MQGMO_BOTE NEXT od té doby, kdy byla fronta otevřena.

Odebrání zprávy, kterou jste procházeli

Z fronty můžete odebrat zprávu, kterou jste již prohlédl, za předpokladu, že jste frontu otevřeli pro odebrání zpráv a také pro procházení. (Je třeba určit jednu z voleb MQOO_INPUT_ * a také volbu MQOO_BROWSE v rámci volání MQOPEN.)

Chcete-li zprávu odebrat znovu, obraťte se na příkaz MQGET znovu, ale v poli *Options* struktury MQGMO určete MQGMO_MSG_UNDER_CURSOR.V takovém případě volání MQGET ignoruje pole *MsgId*, *CorrelId*a *GroupId* struktury MQMD.

V době mezi jednotlivými kroky při procházení a odebrání může jiný program odebírat zprávy z fronty, včetně zprávy pod kurzorem procházení. V takovém případě volání MQGET vrátí kód příčiny, který říká, že zpráva není k dispozici.

Procházení zpráv v logickém pořadí

"Logické a fyzické uspořádání" na stránce 236 vysvětluje rozdíl mezi logickým a fyzickým pořadím zpráv ve frontě. Tento rozdíl je zvláště důležitý při procházení fronty, protože obecně se zprávy neodstraňují a operace procházení nemusí nutně začínat na začátku fronty.

Pokud některá aplikace prochází různými zprávami jedné skupiny (s použitím logického pořadí), je důležité, aby se za účelem dosažení začátku další skupiny mělo následovat logické pořadí, protože poslední zpráva jedné skupiny se může stát fyzicky *po* první zprávě další skupiny. Volba MQGMO_LOGICAL_ORDER zajišťuje, aby při skenování fronty bylo dodržováno logické pořadí.

Použijte funkci MQGMO_ALL_MSGS_AVAILABLE (nebo MQGMO_ALL_SEGMENTS_AVAILABLE) s péčí o operace procházení. Zvažte případ logických zpráv s MQGMO_ALL_MSGS_AVAILABLE. Výsledkem je, že logická zpráva je k dispozici pouze v případě, že jsou přítomny také všechny zbývající zprávy ve skupině. Pokud tomu tak není, zpráva se předá znovu. To může znamenat, že když se chybějící zprávy dostaví později, nejsou při procházení zaznamenány operací procházení.

Jsou-li například přítomny následující logické zprávy,

```
   Logical message 1 (not last) of group 123
   Logical message 1 (not last) of group 456
   Logical message 2 (last)     of group 456
```

a funkce procházení je vydána s parametrem MQGMO_ALL_MSGS_AVAILABLE, je vrácena první logická zpráva skupiny 456 a ponechá se kurzor procházení této logické zprávy. Pokud je nyní doručena druhá (poslední) zpráva skupiny 123:

```
   Logical message 1 (not last) of group 123
   Logical message 2 (last)     of group 123
   Logical message 1 (not last) of group 456  <=== browse cursor
   Logical message 2 (last)     of group 456
```

a je vydána stejná funkce browse-next, nevšiml si, že skupina 123 je nyní dokončena, protože první zpráva této skupiny je *před* kurzorem procházení.

V některých případech (například, pokud jsou zprávy načítány destruktivně, když je skupina přítomna v celém rozsahu), můžete použít MQGMO_ALL_MSGS_AVAILABLE spolu s MQGMO_BROFIRST. Jinak musíte zopakovat procházení procházení, aby bylo možné vzít na vědomí nově přijaté zprávy, které byly vynechány; pouze zadání MQGMO_WAIT spolu s MQGMO_BE NEXT a MQGMO_ALL_MSGS_AVAILABLE nebude brát v úvahu jejich počet. (To se také děje s vysoce prioritními zprávami, které mohou přicházet po skenování zpráv, je dokončeno.)

Další sekce se zabývají procházení příkladů, které se zabývají nesegmentovanými zprávami; segmentované zprávy dodržují podobné zásady.

Procházení zpráv ve skupinách
V tomto příkladu prochází aplikace přes každou zprávu ve frontě v logickém pořadí.

Zprávy ve frontě mohou být seskupeny. Pro seskupené zprávy aplikace nechce spustit zpracování žádné skupiny, dokud nedorazily všechny zprávy v ní obsažené. Funkce MQGMO_ALL_MSGS_AVAILABLE je proto určena pro první zprávu ve skupině; pro následující zprávy ve skupině je tato volba zbytečná.

Funkce MQGMO_WAIT se v tomto příkladu používá. Přestože však lze čekat na doručení nové skupiny z důvodů v produktu “Procházení zpráv v logickém pořadí” na stránce 262, není splněna, pokud kurzor procházení již prošel první logickou zprávou ve skupině, a zbývající zprávy jsou nyní doručeny. Nicméně čekání na vhodný interval zajišťuje, že aplikace nebude při čekání na nové zprávy nebo segmenty neustále smyčkat.

Funkce MQGMO_LOGICAL_ORDER se používá v celém rozsahu, aby bylo zaručeno, že skenování je v logickém pořadí. Tento rozdíl je v rozporu s destruktivním parametrem MQGET, kde je odebíraná každá skupina, MQGMO_LOGICAL_ORDER, při hledání první (nebo jediné) zprávy ve skupině.

Předpokládá se, že vyrovnávací paměť aplikace je vždy dostatečně velká, aby udržela celou zprávu, ať už byla zpráva segmentována, či nikoli. MQGMO_COMPLETE_MSG je proto určen pro každý příkaz MQGET.

Níže je uveden příklad procházení logických zpráv ve skupině:

```
/* Browse the first message in a group, or a message not in a group */
GMO.Options = MQGMO_BROWSE_NEXT | MQGMO_COMPLETE_MSG | MQGMO_LOGICAL_ORDER
            | MQGMO_ALL_MSGS_AVAILABLE | MQGMO_WAIT
MQGET GMO.MatchOptions = MQMO_MATCH_MSG_SEQ_NUMBER, MD.MsgSeqNumber = 1
/* Examine first or only message */
 ...

GMO.Options = MQGMO_BROWSE_NEXT | MQGMO_COMPLETE_MSG | MQGMO_LOGICAL_ORDER
do while ( GroupStatus == MQGS_MSG_IN_GROUP )
   MQGET
   /* Examine each remaining message in the group */
   ...
```

Skupina se opakuje, dokud není navrácena hodnota MQRC_NO_MSG_AVAILABLE.

Procházení a načítání destruktivně
V tomto příkladu aplikace prochází každou z logických zpráv v rámci skupiny před tím, než se rozhodne, zda tuto skupinu destruktivně načíst.

První část tohoto příkladu je podobná předchozí části. Avšak v tomto případě, když jsme procházeli celou skupinou, rozhodli jsme se vrátit a získat ji destruktivním způsobem.

Protože každá skupina je v tomto příkladu odstraněna, MQGMO_LOGICAL_ORDER se nepoužije při hledání první nebo jediné zprávy ve skupině.

Zde je uveden příklad procházení a následné načtení destruktivně:

```
GMO.Options = MQGMO_BROWSE_NEXT | MQGMO_COMPLETE_MSG | MQGMO_LOGICAL_ORDER
            | MQGMO_ALL_MESSAGES_AVAILABE | MQGMO_WAIT
do while ( GroupStatus == MQGS_MSG_IN_GROUP )
   MQGET
   /* Examine each remaining message in the group (or as many as
      necessary to decide whether to get it destructively) */
   ...

if ( we want to retrieve the group destructively )

   if ( GroupStatus == ' ' )
      /* We retrieved an ungrouped message */
      GMO.Options = MQGMO_MSG_UNDER_CURSOR | MQGMO_SYNCPOINT
      MQGET GMO.MatchOptions = 0
      /* Process the message */
      ...

   else
      /* We retrieved one or more messages in a group.  The browse cursor */
      /* will not normally be still on the first in the group, so we have */
      /* to match on the GroupId and MsgSeqNumber = 1.                    */
      /* Another way, which works for both grouped and ungrouped messages,*/
      /* would be to remember the MsgId of the first message when it was  */
      /* browsed, and match on that.                                      */
      GMO.Options = MQGMO_COMPLETE_MSG | MQGMO_SYNCPOINT
      MQGET GMO.MatchOptions = MQMO_MATCH_GROUP_ID
                             | MQMO_MATCH_MSG_SEQ_NUMBER,
           (MQMD.GroupId      = value already in the MD)
            MQMD.MsgSeqNumber = 1
      /* Process first or only message */
      ...

      GMO.Options = MQGMO_COMPLETE_MSG | MQGMO_SYNCPOINT
                  | MQGMO_LOGICAL_ORDER
      do while ( GroupStatus == MQGS_MSG_IN_GROUP )
         MQGET
         /* Process each remaining message in the group */
         ...
```

Vyvarování se opakovaného doručení procházených zpráv

Pomocí určitých voleb otevření a voleb pro získání zprávy můžete označit zprávy jako procházené tak, aby nebyly znovu načteny aktuální nebo jinými spolupracujícími aplikacemi. Zprávy mohou být explicitně nebo automaticky označeny jako nové, aby je bylo možné znovu zpřístupnit pro prohlížení.

Pokud procházíte zprávy ve frontě, můžete je načíst v jiném pořadí, než je pořadí, ve kterém byste je získali, pokud jste je získali destruktivně. Zejména můžete několikrát procházet stejnou zprávu, která není možná, pokud byla odebrána z fronty. Chcete-li se této zprávě vyhnout, můžete *označit* zprávy při jejich prohlížení a zabránit načítání označených zpráv. Toto je někdy označováno jako *procházení s označením*. Chcete-li označit procházené zprávy, použijte volbu get message MQGMO_MARKWSE_HANDLE a k načtení pouze zpráv, které nejsou označeny, použijte MQGMO_UNMARKED_BERWSE_MSG. Pokud použijete kombinaci voleb MQGMO_BROFIRST FIRST, MQGMO_UNMARKED_BERWSE_MSG a MQGMO_MARK_BROWSE_HANDLE a zadáte opakované volání MQGET, budete každou zprávu ve frontě postupně načítán. Tím se zabrání opakovanému doručení zpráv i v případě, že operace MQGMO_Bated FIRST se používá k zajištění toho, že zprávy nebudou vynechány. Tato kombinace voleb může být reprezentována jedinou konstantou MQGMO_BROWSE_HANDLE. Pokud ve frontě nejsou žádné zprávy, které nebyly procházeny, je vrácen objekt MQRC_NO_MSG_AVAILABLE.

Pokud prohledávaní více aplikací prochází stejnou frontu, mohou otevřít frontu s volbami MQOO_CO_OP a MQOO_BROWSE. Manipulátor objektu vrácený jednotlivými MQOPEN je považován za součást spolupracující skupiny. Jakákoli zpráva vrácená voláním MQGET s uvedením volby MQGMO_MARK_BROWSE_CO_OP je považována za označenou pro tuto spolupracující sadu manipulátorů.

Je-li zpráva již nějakou dobu označena, může ji správce front automaticky zrušit a zpřístupnit k jeho opětovnému prohlížení. Atribut správce front MsgMarkBrowseInterval udává dobu v milisekundách, po kterou má zpráva zůstat označena pro spolupracující sadu manipulátorů. MsgMarkBrowseInterval z -1 znamená, že zprávy nejsou nikdy automaticky neoznačené.

Když se zastaví jeden proces nebo sada procesů pro spolupráci značení zpráv, všechny označené zprávy budou neoznačeny.

Příklady spolupráce při procházení

Můžete spustit více kopií aplikace dispečera k procházení zpráv ve frontě a zahájení odběratele na základě obsahu každé zprávy. V každém dispečeru otevřete frontu s MQOO_CO_OP. To znamená, že dispečery spolupracují a budou si být vědomi, že se jedná o označené zprávy. Každý dispečer poté provádí opakované volání MQGET, přičemž určuje volby MQGMO_BROTE_FIRST, MQGMO_UNMARKED_BROWSE_MSG a MQGMO_MARK_BROWSE_CO_OP (můžete použít jedinou konstantu MQGMO_BROWSE_CO_OP, která představuje tuto kombinaci voleb). Každá dispečerské aplikace pak načte pouze ty zprávy, které ještě nebyly označeny jinými spolupracujícími dispečery. Dispečer inicializuje spotřebitele a předává prvek MsgToken vrácený procesem MQGET pro spotřebitele, který destruktivně získá zprávu z fronty. Pokud spotřebitel zazálohuje příkaz MQGET zprávy, je zpráva k dispozici pro jeden z prohlížečů k opětnému odeslání, protože již není označena. Pokud spotřebitel neprovede příkaz MQGET ve zprávě, poté, co byl předán objekt MsgMarkBrowseInterval , správce front zruší označení zprávy pro spolupracující sadu manipulátorů a lze ji znovu odbavit.

Spíše než více kopií stejné aplikace dispečeru, můžete mít několik různých aplikací dispečera prohledávajících frontu, z nichž každá je vhodná pro zpracování podmnožiny zpráv ve frontě. V každém dispečeru otevřete frontu s MQOO_CO_OP. To znamená, že dispečery spolupracují a budou si být vědomi, že se jedná o označené zprávy.

- Je-li pro jeden dispečer důležité pořadí zpracování zpráv, každý dispečer provádí opakované volání MQGET, přičemž určuje volby MQGMO_BROTE_FIRST, MQGMO_UNMARKED_BROWSE_MSG a MQGMO_MARK_BROWSE_HANDLE (nebo MQGMO_BROWSE_HANDLE). Je-li procházená zpráva vhodná pro tento dispečer ke zpracování, potom provede volání MQGET s parametrem MQMO_MATCH_MSG_TOKEN, MQGMO_MARK_BROWSE_CO_OP a MsgToken vráceným předchozím voláním MQGET. Je-li volání úspěšné, dispečer inicializuje spotřebitele a předá mu MsgToken .
- Není-li pořadí zpracování zpráv důležité a očekává se, že dispečer zpracuje většinu zpráv, které zjistí, použijte volby MQGMO_BROTE_FIRST, MQGMO_UNMARKED_BERWSE_MSG a MQGMO_MARK_BROWSE_CO_OP (nebo MQGMO_BROWSE_CO_OP). Pokud dispečer prohlíží zprávu, kterou nemůže zpracovat, zruší označení této zprávy voláním příkazu MQGET s volbou MQMO_MATCH_MSG_TOKEN, MQGMO_UNMARK_BROWSE_CO_OP a MsgToken vráceným dříve.

Některé případy, kdy se volání MQGET nezdaří

Změní-li se některé atributy fronty použitím volby FORCE u příkazu mezi zadáním volání MQOPEN a MQGET, volání MQGET selže a vrátí kód příčiny MQRC_OBJECT_CHANGED.

Správce front označí obslužnou rutinu objektu jako neplatnou. K tomu dojde také v případě, že se změny použijí na každou frontu, na kterou je název fronty vyřešen. Atributy, které ovlivňují popisovač tímto způsobem, jsou uvedeny v popisu volání MQOPEN v MQOPEN. Pokud vaše volání vrátí kód příčiny MQRC_OBJECT_CHANGED, zavřete frontu a znovu ji otevřete a poté se pokuste zprávu znovu získat.

Pokud operace get jsou blokovány pro frontu, ze které se pokoušíte získat zprávy (nebo frontu, na kterou má být název fronty rozpoznán), volání MQGET selže a vrátí kód příčiny MQRC_GET_INHIBITED. K tomu dojde dokonce i v případě, že používáte volání MQGET pro procházení. Možná budete moci úspěšně získat zprávu, pokud se pokusíte o volání MQGET později, je-li návrh aplikace takový, že jiné programy pravidelně mění atributy front.

Byla-li dynamická fronta (dočasná nebo trvalá) odstraněna, volání MQGET s použitím dříve získaného manipulátoru objektu selžou a vrátí kód příčiny MQRC_Q_DELETED.

Zapisování aplikací typu publikování/odběr

Začněte zapisovat do aplikací WebSphere MQ publish/subscribe.

Přehled konceptů publikování/odběru viz Úvod do systému zpráv publikování a odběru produktu WebSphere MQ.

Informace o psaní různých typů aplikací pro publikování/odběr naleznete v následujících tématech:

- “Zapisování aplikací vydavatele” na stránce 266
- “Zapisování aplikací odběratele” na stránce 273
- “Životní cykly publikování/odběru” na stránce 290
- “Vlastnosti zprávy publikování/odběru” na stránce 295
- “Uspořádání zpráv” na stránce 296
- “Zachycení publikací” na stránce 297
- “Volby publikování” na stránce 305
- “Volby odběru” na stránce 305

Související pojmy

“Koncepty vývoje aplikací” na stránce 7
K zápisu aplikací IBM WebSphere MQ můžete použít volbu procedurálních nebo objektově orientovaných jazyků. Informace o konceptech produktu IBM WebSphere MQ , které jsou užitečné pro vývojáře aplikací, naleznete v odkazech v tomto tématu.

“Rozhodování o tom, jaký programovací jazyk použít” na stránce 75
Tyto informace použijte k vyhledání informací o programovacích jazycích a rámcích podporovaných produktem IBM WebSphere MQa o některých aspektech jejich použití.

“Návrh aplikací produktu IBM WebSphere MQ” na stránce 86
Když se rozhodnete, jak aplikace mohou využívat výhody platforem a prostředí, které máte k dispozici, musíte se rozhodnout, jak budete používat funkce nabízené produktem WebSphere MQ.

“Ukázka programů WebSphere MQ” na stránce 92
Tato kolekce témat slouží k získání informací o ukázkových programech WebSphere MQ na různých platformách.

“Psaní front aplikace” na stránce 187
Prostřednictvím těchto informací můžete získat informace o vytváření aplikací pro řazení do front, připojování a odpojování od správce front, publikování a odběru a otevírání a zavírání objektů.

“Tvorba klientských aplikací” na stránce 337
Co potřebujete vědět, chcete-li zapisovat klientské aplikace do produktu WebSphere MQ.

“Použití webových služeb v produktu WebSphere MQ” na stránce 913
Můžete vyvíjet aplikace produktu IBM WebSphere MQ pro webové služby pomocí přenosu IBM WebSphere MQ pro protokol SOAP nebo mostu produktu IBM WebSphere MQ pro protokol HTTP.

“Sestavení aplikace IBM WebSphere MQ” na stránce 412
Tyto informace použijte k seznámení se s aplikací produktu IBM WebSphere MQ na různých platformách.

“Obsluha chyb programu” na stránce 529
Tyto informace vysvětlují chyby související s voláními MQI MQI buď při volání, nebo při doručení její zprávy do konečného cíle.

Zapisování aplikací vydavatele

Začněte pracovat se zápisem aplikací vydavatele tak, že prostudujete dva příklady. První je modelován co nejpřesněji v okamžiku, kdy aplikace boduje aplikaci do fronty, a druhý demonstruje vytváření témat dynamicky-běžnější vzor pro aplikace vydavatele.

Zapsání jednoduché aplikace WebSphere MQ Publisher je stejně jako zápis do aplikace bodu WebSphere MQ , která umísťuje zprávy do fronty (Tabulka 41 na stránce 267). Rozdíl spočívá v tom, že se zprávy MQPUT změní na téma, ne do fronty.

Tabulka 41. Vzor bodu k bodu versus publikování/odběr schématu programu WebSphere MQ .

Krok	**Bod připojení k bodu MQ**	**Publikovat volání MQ**
Připojit se ke správci front	MQCONN	MQCONN
Otevřít frontu	MQOPEN	
Otevřít téma		MQOPEN
Vložit zprávu (y)	MQPUT	MQPUT
Zavřít téma		MQCLOSE
Uzavřít frontu	MQCLOSE	
Odpojit od správce front	MQDISC	MQDISC

K tomu, aby se tento beton, existují dva příklady žádostí o zveřejnění cen akcií. V prvním příkladu (“Příklad 1: Vydavatel na pevné téma” na stránce 267), který je podrobně modelován při vkládání zpráv do fronty, vytvoří administrátor definici tématu podobným způsobem, jak vytvořit frontu. Programátorské kódy MQPUT slouží k zápisu zpráv na téma místo jejich zápisu do fronty. Ve druhém příkladě (“Příklad 2: Vydavatel na téma proměnné” na stránce 270) je vzorec interakce programu s produktem WebSphere MQ podobný. Rozdíl je v tom, že programátor poskytuje téma, do něhož je zpráva zapisována, spíše než administrátor. V praxi to obvykle znamená, že řetězec tématu je definován jako obsah nebo je poskytován jiným zdrojem, jako např. lidským vstupem prostřednictvím prohlížeče.

Související pojmy

“Zapisování aplikací odběratele” na stránce 273
Začínáme se zapisováním aplikací odběratele tím, že studujete tři příklady: aplikace WebSphere MQ spotřebovávající zprávy z fronty, aplikace, která vytváří odběr a nevyžaduje žádné znalosti o řazení do front, a konečně příklad, který používá řazení do front i odběry.

Související odkazy

DEFINOVAT TÉMA
ZOBRAZIT TÉMA
ZOBRAZIT STAV TPSTATUS

Příklad 1: Vydavatel na pevné téma

Program WebSphere MQ pro ilustraci publikování na administrativně definované téma.

Poznámka: Styl kompaktního kódování je určen pro přehlednost, nikoli pro použití v produkčním prostředí.

Viz výstup v části Obrázek 38 na stránce 268

```
#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 37. Jednoduchý vydavatel WebSphere MQ pro určité 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 38. Ukázkový výstup z příkladu prvního vydavatele

Následující vybrané řádky kódu popisují aspekty psaní aplikace vydavatele pro produkt WebSphere MQ.

char topicNameDefault[] = "IBMSTOCKPRICE";
V programu je definován výchozí název tématu. Můžete ji přepsat zadáním názvu jiného objektu tématu jako prvního argumentu pro program.

MQCHAR resTopicStr[151];
`resTopicStr` je uveden v `td.ResObjectString.VSPtr` a je používán produktem `MQOPEN` k vrácení vyřešeného řetězce tématu. Učinit délku `resTopicStr` o jednu větší než délku předanou v produktu `td.ResObjectString.VSBufSize` za účelem poskytnutí prostoru pro ukončení s hodnotou null.

memset (resTopicStr, 0, sizeof(resTopicStr));
Inicializujte `resTopicStr` na hodnotu null, abyste zajistili, že přeložený řetězec tématu vrácený v `MQCHARV` má hodnotu null ukončen.

td.ObjectType = MQOT_TOPIC
Pro publikování/odběr je k dispozici nový typ objektu: *objekt tématu*.

td.Version = MQOD_VERSION_4;
Chcete-li použít nový typ objektu, musíte použít alespoň *verzi 4* deskriptoru objektu.

strncpy(td.ObjectName, topicName, MQ_OBJECT_NAME_LENGTH);
`topicName` je název objektu tématu, který se někdy označuje jako objekt administrativního tématu. V tomto příkladu je třeba objekt tématu vytvořit předem pomocí Průzkumníka WebSphere MQ nebo tohoto příkazu MQSC,

```
DEFINE TOPIC(IBMSTOCKPRICE) TOPICSTR(NYSE/IBM/PRICE) REPLACE;
```

td.ResObjectString.VSPtr = resTopicStr;
Vyřešený řetězec tématu se vypisuje v konečném `printf` v programu. Nastavte strukturu `MQCHARV` `ResObjectString` pro produkt WebSphere MQ tak, aby vracel přeložený řetězec zpět do programu.

MQOPEN(Hconn, &td, MQOO_OUTPUT | MQOO_FAIL_IF_QUIESCING, &Hobj, &CompCode, &Reason);
Otevřít téma pro výstup; stejně jako otevření fronty pro výstup.

pmo.Options = MQPMO_FAIL_IF_QUIESCING | MQPMO_RETAIN;
Chcete, aby byli noví odběratelé schopni přijímat publikování, a uvedením `MQPMO_RETAIN` v vydavateli, když spustíte odběratele, obdrží nejnovější publikování, publikovaná před spuštěním odběratele, jako první odpovídající publikování. Alternativou je poskytnout odběratelům publikování publikovaná pouze poté, co je odběratel spuštěn. Kromě toho má odběratel možnost odmítnout přijetí zachovaného publikování zadáním `MQSO_NEW_PUBLICATIONS_ONLY` ve svém odběru.

MQPUT(Hconn, Hobj, &md, &pmo, (MQLONG)strlen(publication)+1, publication, &CompCode, &Reason);
Přidejte 1 až do délky řetězce předaného do `MQPUT` , čímž předáváte znak ukončení null do WebSphere MQ jako část vyrovnávací paměti zpráv.

Co první příklad demonstruje? Příklad imituje co nejpřesněji vyzkoušený a testovaný tradiční vzor pro zápis programů do bodu WebSphere MQ . Důležitým rysem programovacího modelu produktu WebSphere MQ je to, že programátor není znepokojen tím, že jsou odesílány zprávy. Úkolem programátora je připojit se ke správci front a předat mu zprávy, které mají být distribuovány příjemcům. V paradigmatu mezi dvěma body programátor otevře frontu (pravděpodobně alias frontu alias), kterou administrátor nakonfiguroval. Fronta aliasů směruje zprávy do cílové fronty, a to buď na lokálním správci front, nebo na vzdáleném správci front. Zatímco zprávy čekají na doručení, jsou uloženy ve frontách někde mezi zdrojem a místem určení.

Místo otevření fronty otevře programátor v rámci vzoru publikování/odběru téma. V našem příkladu je téma asociováno s řetězcem tématu administrátorem. Správce front předá publikování s použitím front lokálním nebo vzdáleným odběratelům, kteří mají odběry odpovídající řetězci tématu publikování. Pokud jsou publikace uchovávány, uchovává správce front nejnovější kopii této publikace, a to i v případě, že dosud nemá žádné odběratele. Zachované publikování je k dispozici pro postoupení budoucím odběratelům. Aplikace vydavatele nehraje žádnou roli při výběru nebo směrování publikování do místa určení; jejím úkolem je vytvářet a vkládat publikace do témat definovaných administrátorem.

Tento příklad fixního tématu je atypický pro mnoho aplikací typu publikování/odběr: je statický. Vyžaduje to, aby administrátor definoval řetězce témat a změnil témata, která jsou publikována. Aplikace publish/subscribe publish-subscribe musí znát některé nebo všechny stromy témat. Možná se témata často mění, nebo snad i když se témata příliš nemění, počet kombinací témat je velký a pro administrátora je příliš obtížné definovat uzel tématu pro každý řetězec tématu, který může být potřeba publikovat. Možná řetězce tématu nejsou známy v předstihu publikování; aplikace vydavatele může použít informace z obsahu publikování k určení řetězce tématu, nebo může obsahovat informace o řetězcích témat pro publikování z jiného zdroje, jako je například vstup z lidských zdrojů z prohlížeče. Následující příklad ukazuje, jak dynamicky vytvářet témata jako součást aplikace vydavatele k zajišťování dynamičtějších stylů publikování.

Témata, vydavatelé a odběratelé dohromady. Navrhování pravidel nebo architektury pro pojmenovávání témat a jejich uspořádání do stromů témat je důležitým krokem při vývoji řešení typu publikování-odběr. Pečlivě se podívejte na to, do jaké míry se organizace stromu témat spojí s vydavatelem a programy odběratele dohromady a spojí je s obsahem stromu témat. Zeptejte se sami sebe na otázku, zda změny ve stromu témat ovlivňují vydavatele a aplikace odběratele, a jak můžete minimalizovat efekt. Zabudovaná do architektury modelu WebSphere MQ publish/subscribe je pojem objektu administrativního tématu, který poskytuje základní část, neboli kořenový podstrom, tématu. Objekt tématu vám dává možnost definovat kořenovou část stromu témat administrativně, která zjednodušuje programování a provoz aplikací, a následně vylepšuje udržovatelnost. Pokud například implementujete více aplikací publikování/odběru, které mají izolované stromy témat, pak administrativně definováním kořenové části stromu témat, můžete zaručit izolaci stromu témat i v případě, že neexistuje konzistence v konvencích pojmenování tématu, které byly přijaty různými aplikacemi.

V praxi se aplikace vydavatele týkají spektra výhradně pomocí pevných témat, podobně jako v tomto příkladu, a proměnných témat, jako v následujícím příkladu. Produkt “Příklad 2: Vydavatel na téma proměnné” na stránce 270 také demonstruje kombinaci použití témat a řetězců témat.

Související pojmy

“Příklad 2: Vydavatel na téma proměnné” na stránce 270
Program Websphere MQ pro ilustraci publikování na téma definované programem.

“Zapisování aplikací odběratele” na stránce 273
Začínáme se zapisováním aplikací odběratele tím, že studujete tři příklady: aplikace WebSphere MQ spotřebovávající zprávy z fronty, aplikace, která vytváří odběr a nevyžaduje žádné znalosti o řazení do front, a konečně příklad, který používá řazení do front i odběry.

Příklad 2: Vydavatel na téma proměnné

Program Websphere MQ pro ilustraci publikování na téma definované programem.

Poznámka: Styl kompaktního kódování je určen pro přehlednost, nikoli pro použití v produkčním prostředí.

Viz výstup v části Obrázek 40 na stránce 272.

```
#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 39. Jednoduchý vydavatel WebSphere MQ k tématu s proměnnými.

```
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 40. Příklad výstupu ukázky z druhého vydavatele

Je zde několik poznámek k tomuto příkladu.

char topicNameDefault[] = "STOCKS";
Výchozí název tématu STOCKS definuje část řetězce tématu. Tento název tématu můžete přepsat tak, že jej poskytnete jako první argument programu, nebo jej odstraňte zadáním parametru / jako prvního parametru.

char topicString[101] = "IBM/PRICE";
IBM/PRICE je výchozí řetězec tématu. Tento řetězec tématu můžete přepsat tím, že jej poskytnete jako druhý argument programu.

Správce front kombinuje řetězec tématu poskytovaný objektem tématu STOCKS , "NYSE", s řetězcem tématu poskytnutým programem "IBM/PRICE" a vkládá mezi tyto dva řetězce témat "/" . Výsledkem je vyřešený řetězec tématu "NYSE/IBM/PRICE". Výsledný řetězec tématu je stejný jako řetězec definovaný v objektu tématu produktu IBMSTOCKPRICE a má přesně stejný účinek.

Objekt administrativního tématu přidružený k vyřešenému řetězci tématu nemusí být nutně totožný s objektem tématu, který byl předán vydavateli MQOPEN . Produkt WebSphere MQ pomocí stromu implicitního v analyzovaného řetězce tématu odpracuje, který objekt administrativního tématu definuje atributy přidružené k této publikaci.

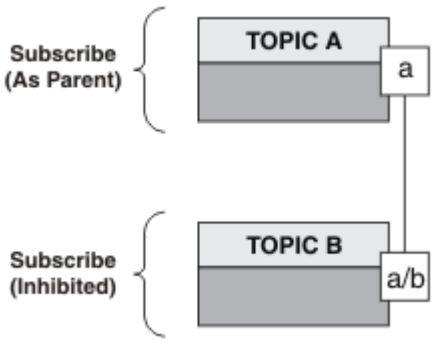
Předpokládejme, že existují dva objekty témat A a B, a A definuje téma "a"a B definuje téma "a/b" (Obrázek 41 na stránce 273). Pokud se program vydavatele odkazuje na objekt tématu A a poskytuje řetězec tématu "b", interpretujete téma na řetězec tématu "a/b", pak publikování dědí vlastnosti z objektu tématu B , protože téma se shoduje s řetězcem tématu "a/b" definovaným pro B.

if (strcmp(argv[1],"/"))
argv[1] je volitelně zadán parametr topicName. "/" je neplatný jako název tématu; zde označuje, že neexistuje žádné jméno tématu, a řetězec tématu je poskytován zcela programem. Výstup v produktu Obrázek 40 na stránce 272 zobrazuje řetězec celého tématu dynamicky dodávaný programem.

strncpy(td.ObjectName, topicName, MQ_OBJECT_NAME_LENGTH);
V případě výchozího případu je třeba předem vytvořit volitelný topicName pomocí Průzkumníka WebSphere MQ nebo pomocí tohoto příkazu MQSC:

```
DEFINE TOPIC(STOCKS) TOPICSTR(NYSE) REPLACE;
```

td.ObjectString.VSPtr = topicString;
Řetězec tématu je pole MQCHARV v deskriptoru tématu



Obrázek 41. Přidružení objektů témat

Co druhý příklad demonstruje? Ačkoli je kód velmi podobný prvnímu příkladu-efektivně existují pouze dva řádky-výsledek je výrazně odlišný od prvního programu. Programátor určuje místa určení, do kterých jsou odesílána publikace. Ve spojení s minimálním vstupem administrátora použitým k návrhu aplikací odběratele, není třeba předem definovat žádná témata nebo fronty pro směrování publikací od vydavatelů k odběratelům.

V paradigmatu systému zpráv typu point-to-point je třeba definovat fronty před tím, než je možné zprávy směrovat. Pro publish/subscribe to neplatí, ačkoli produkt WebSphere MQ implementuje publikování/odběr pomocí systému pro ukládání dat do fronty; výhody garantovaného doručení, transakcese a volné vazby přidružené k systému zpráv a zařazování do front jsou zděděny prostřednictvím aplikací publikování/odběru.

Vývojář musí rozhodnout, zda vydavatel a odběratel mají být informováni o základním stromu témat nebo ne, a také zda si programy odběratele vědí, zda jsou zařazovány do fronty nebo ne. Studium ukázkových aplikací odběratele další. Jsou navrženy tak, aby byly používány s příklady vydavatele, obvykle publikováním a přihlášením k odběru `NYSE/IBM/PRICE`.

Související pojmy

“Příklad 1: Vydavatel na pevné téma” na stránce 267
Program WebSphere MQ pro ilustraci publikování na administrativně definované téma.

“Zapisování aplikací odběratele” na stránce 273
Začínáme se zapisováním aplikací odběratele tím, že studujete tři příklady: aplikace WebSphere MQ spotřebovávající zprávy z fronty, aplikace, která vytváří odběr a nevyžaduje žádné znalosti o řazení do front, a konečně příklad, který používá řazení do front i odběry.

Zapisování aplikací odběratele

Začínáme se zapisováním aplikací odběratele tím, že studujete tři příklady: aplikace WebSphere MQ spotřebovávající zprávy z fronty, aplikace, která vytváří odběr a nevyžaduje žádné znalosti o řazení do front, a konečně příklad, který používá řazení do front i odběry.

V produktu Tabulka 42 na stránce 274 jsou uvedeny tři styly odběratele nebo odběratele, společně s posloupnostmi funkcí WebSphere MQ , které je charakterizují.

1. První styl MQ Publication Consumer je identický s bodem k bodu MQ , který provádí pouze MQGET. Aplikace nemá žádné informace o tom, že se jedná o využití publikací-stačí číst zprávy z fronty. Odběr, který způsobí přesměrování publikování do fronty, je vytvořen administrativně pomocí programu Průzkumník produktu WebSphere MQ nebo pomocí příkazu.
2. Druhý styl je upřednostňovaným vzorem pro většinu aplikací odběratele. Aplikace odběratele vytvoří odběr a poté získá publikace. Správa front je prováděna správcem front.
3. Ve třetím stylu se aplikace odběratele rozhodne otevřít a zavřít základní frontu, která se používá pro publikování, a také vydávat odběry pro vyplnění fronty s publikacemi.

Jedním ze způsobů, jak porozumět těmto stylům, je prostudovat si vzorové programy C uvedené v Tabulka 42 na stránce 274 pro každý ze stylů. Příklady jsou navrženy ke spuštění ve spojení s příkladem vydavatele nalezeným v souboru "Zapisování aplikací vydavatele" na stránce 266 .

Tabulka 42. Vzory programu WebSphere MQ Point to point vs. subscribe.

Krok	Spotřebitel zpráv MQ	"Příklad 1: Spotřebitel publikování MQ" na stránce 274	"Příklad 2: Spravovaný odběratel produktu MQ" na stránce 277	"Příklad 3: Nespravovaný odběratel MQ" na stránce 282
Připojit se ke správci front	MQCONN	MQCONN	MQCONN	MQCONN
Otevřít frontu	MQOPEN	MQOPEN		MQOPEN
Odebírat			MQSUB	MQSUB
Získat zprávu (y)	MQGET	MQGET	MQGET	MQGET
Uzavřít frontu	MQCLOSE	MQCLOSE	(MQCLOSE)	MQCLOSE
Zavřít odběr			MQCLOSE	MQCLOSE
Odpojit od správce front	MQDISC	MQDISC	MQDISC	MQDISC

Použití funkce MQCLOSE je vždy volitelné, buď pro uvolnění prostředků, předání voleb MQCLOSE nebo pouze pro symetrii s MQOPEN. Vzhledem k tomu, že není pravděpodobné, že je třeba určit volby MQCLOSE, je-li fronta odběru uzavřena v případu odběratele spravovaných MQ a argument symetrie není relevantní, fronta odběru není v produktu Příklad 2: Spravovaný odběratel produktu MQ explicitně uzavřena.

Jiný způsob, jak porozumět vzorům aplikací publikování/odběru, je také příliš pohledem na interakce mezi různými zúčastněnými objekty. Linie, nebo sekvenční diagramy UML jsou dobrým způsobem ke studiu interakcí. Tři příklady linie linie jsou popsány v "Životní cykly publikování/odběru" na stránce 290.

Příklad 1: Spotřebitel publikování MQ

Spotřebitel zpráv produktu MQ je spotřebitel zpráv produktu IBM WebSphere MQ , který se nepřihlašují k odběru vlastních témat.

Chcete-li vytvořit odběr a frontu publikování pro tento příklad, spusťte následující příkazy nebo definujte objekty pomocí Průzkumníka produktu WebSphere MQ .

```
DEFINE QLOCAL(STOCKTICKER) REPLACE;
DEFINE SUB(IBMSTOCKPRICESUB) DEST(STOCKTICKER) TOPICOBJ(IBMSTOCKPRICE) REPLACE;
```

Odběr produktu `IBMSTOCKPRICESUB` se odkazuje na objekt tématu produktu `IBMSTOCK` vytvořený pro příklad vydavatele a na lokální frontu `STOCKTICKER`. Objekt tématu `IBMSTOCK` definuje řetězec tématu, který se používá v odběru `NYSE/IBM/PRICE`. Všimněte si, že objekt tématu a fronta použitá pro příjem publikování musí být definována před vytvořením odběru.

Šablona spotřebitele publikování MQ obsahuje mnoho cenných fazet:

1. Vícenásobné zpracování: sdílení mimo práci na čtení publikací. Všechny publikace se nacházejí v jedné frontě přidružené k odběru tématu odběru. Frontu s použitím produktu `MQOO_INPUT_SHARED`může otevřít více spotřebitelů.
2. Centrálně spravované odběry. Aplikace nesestavují svá vlastní témata odběru nebo odběry; administrátor je odpovědný za místa odeslání publikací.
3. Spojení odběru: více různých odběrů lze odeslat do jediné fronty.

4. Trvalost odběru: Fronta přijímá všechna publikování bez ohledu na to, zda jsou či nejsou aktivní spotřebitelé.
5. Migrace a koexistence: kód odběratele funguje stejně dobře pro dvoubodový systém a scénář publikování/odběru.

Odběr vytvoří vztah mezi řetězcem tématu `NYSE/IBM/PRICE` a frontou `STOCKTICKER`. Publikace, včetně všech aktuálně zachovaných publikací, jsou od okamžiku vytvoření odběru předány produktu `STOCKTICKER` .

Administrativně vytvořený odběr může být spravován nebo nespravovaný. Spravovaný odběr se projeví, jakmile byl vytvořen, stejně jako nespravovaný odběr. Ne všechny fazety vzorku jsou k dispozici pro spravovaný odběr. Viz “Příklad 3: Nespravovaný odběratel MQ” na stránce 282

Poznámka: Styl kompaktního kódování je určen pro přehlednost, nikoli pro použití v produkčním prostředí.

Výsledky jsou zobrazeny v Obrázek 43 na stránce 276.

```
#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 42. 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 43. Výstup ze spotřebitele publikování MQ

K dispozici je několik standardních rad k programování v jazyku WebSphere 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 1 do `strlen(publication)`. Nastavení vyrovnávacích pamětí MQCHAR na hodnotu null je dobrým programovacím stylem pro programy IBM WebSphere MQ C, které používají vyrovnávací paměti k ukládání řetězců a zajišťují, že hodnota null bude obsahovat pole znaků, které nevyplní vyrovnávací paměť zcela.

MQGET(Hconn, Hobj, &md, &gmo, sizeof(publicationBuffer)-1, publication, &messlen, &CompCode, &Reason);
Vyhrazení jedné hodnoty null na konci vyrovnávací paměti zpráv zajistí, že vrácená zpráva má koncovou hodnotu null v případě, že "`if (messlen == strlen(publication));`" je true. Tento tip doplňuje předchozí a zajišťuje, že v `publicationBuffer` existuje alespoň jedna hodnota null, která není přepsána obsahem produktu `publication`.

Související pojmy

“Příklad 2: Spravovaný odběratel produktu MQ” na stránce 277
Odběratel MQ je upřednostňovaným vzorem pro většinu aplikací odběratele. Příklad vyžaduje *ne* administrative definition of queues, témata nebo .

“Příklad 3: Nespravovaný odběratel MQ” na stránce 282
Nespravovaný odběratel je důležitou třídou aplikace odběratele. S ní kombinujete výhody publikování/odběru s *ovládacím prvkem* řazení do fronty a spotřebou publikací. Tento příklad ukazuje různé způsoby kombinování odběrů a front.

“Zapisování aplikací vydavatele” na stránce 266
Začněte pracovat se zápisem aplikací vydavatele tak, že prostudujete dva příklady. První je modelován co nejpřesněji v okamžiku, kdy aplikace boduje aplikaci do fronty, a druhý demonstruje vytváření témat dynamicky-běžnější vzor pro aplikace vydavatele.

Příklad 2: Spravovaný odběratel produktu MQ

Odběratel MQ je upřednostňovaným vzorem pro většinu aplikací odběratele. Příklad vyžaduje *ne* administrative definition of queues, témata nebo .

Tento nejjednodušší druh spravovaného odběratele obvykle používá odběr *netrvalé* . Tento příklad se zaměřuje na netrvalý odběr. Odběr trvá pouze , pokud je doba trvání popisovače odběru z produktu MQSUB. Všechny publikace , které odpovídají řetězci tématu během doby životnosti odběru, jsou odeslány do fronty odběru (a případně zachované publikace, pokud příznak `MQSO_NEW_PUBLICATIONS_ONLY` není nastaven nebo je nastaven na výchozí hodnotu, předchozí publikování odpovídající řetězci tématu bylo zachováno a publikování bylo trvalé nebo správce front nebyl ukončen, od té doby, co byla publikace vytvořena).

S tímto vzorem můžete také použít *trvalý* odběr. Obvykle je-li použit spravovaný trvalý odběr, a to z důvodů spolehlivosti, spíše než aby byl vytvořen odběr, který by bez výskytu chyb mohl odžít odběratele. Další informace o různých životních cyklech přidružených ke spravovaným, nespravovaným, trvalých a netrvalých odběrů naleznete v sekci souvisejících témat.

Trvalé odběry jsou často přidruženy k trvalým publikacím a netrvalé odběry s netrvalými publikacemi, ale neexistuje žádný nutný vztah mezi trvanlivostí odběru a perzistencí publikování. Všechny čtyři kombinace perzistence a trvanlivosti jsou možné.

Pro spravovaný netrvalý případ bude správce front vytvořit frontu odběru, která je vyprázdněna a odstraněna, když je fronta zavřena. Publikování budou z fronty odebrány při zavření netrvalého odběru.

Cenné fazety spravovaného netrvanlivého vzoru, které jsou příkladem tohoto kódu, jsou následující:

1. Na vyžádání odběru : Řetězec tématu odběru je dynamický. Poskytne ji aplikace, když je spuštěna.
2. Vlastní správa fronty: Fronta odběru je sama definující a spravuje.
3. Vlastní správa životního cyklu odběru: *non-trvalých* odběrů existují pouze po dobu trvání aplikace odběratele.
   - Definujete-li *trvalý* odběr, budou výsledky ve frontě trvalého odběru a publikování uložena na této stránce, přičemž nebudou aktivní žádné programy odběratele. Správce front tuto frontu odstraní (a

vymaže z ní všechny nenačtené publikace) až poté, co se administrátor nebo administrátor zvolí odstranění odběru. Odběr lze odstranit pomocí administrativního příkazu nebo uzavřením odběru s použitím volby MQCO_REMOVE_SUB .

- Uvažte nastavení SubExpiry pro trvalé odběry, takže publikování přestanou být odesílána do fronty a odběratel může spotřebovat všechny zbývající publikace před odebráním odběru a způsobit, že správce front odstraní frontu a všechny zbývající publikace na ní.

4. Flexibilní implementace řetězce témat: Správa témat odběru je zjednodušena definováním základní části odběru pomocí administrativně definovaného tématu. Kořenová část stromu témat je poté skryta z aplikace. Podle skrývání kořenové části lze aplikaci implementovat bez neúmyslného vytvoření stromu témat, který se překrývá s jiným stromem témat vytvořeným jinou instancí, nebo jinou aplikací.
5. Spravovaná témata: od pomocí řetězce tématu, ve kterém první část odpovídá administrativně definovanému objektu tématu, jsou publikace spravovány v souladu s atributy objektu tématu.
   - For example, if the first part of the topic string matches the topic string associated with a clustered topic object, then the subscription can receive publications from other members of the cluster
   - Selektivní shoda administrativně definovaných objektů témat a programově definovaných odběrů umožňuje kombinovat výhody obou. The administrator provides attributes for topics, and the programmer dynamically defines "sub-topics" without being concerned about the management of topics.
   - je výsledný řetězec tématu, který se používá ke shodě s objektem tématu, který poskytuje atributy přidružené k tématu, a nemusí být nutně objekt tématu pojmenovaný v sd.Objectname, ačkoli se obvykle změní na jeden a stejný. Viz “Příklad 2: Vydavatel na téma proměnné” na stránce 270.

Produkt , který provádí trvalý odběr v tomto příkladu, bude nadále odesílán do fronty odběru poté, co odběratel uzavřel odběr s volbou MQCO_KEEP_SUB. Fronta pokračuje v přijímání publikování, pokud není aktivní odběratel. Toto chování můžete potlačit vytvořením odběru s použitím volby MQSO_PUBLICATIONS_ON_REQUEST a použitím produktu MQSUBRQ požadovat zachované publikování.

Odběr lze obnovit později otevřením odběru s použitím volby MQCO_RESUME .

Můžete použít popisovač fronty Hobj, vrácený produktem MQSUB v mnoha ohledech. Popisovač fronty se používá v příkladu k zjišťování názvu fronty odběru. Spravované fronty se otevírají s použitím výchozích modelových front SYSTEM.NDURABLE.MODEL.QUEUE nebo SYSTEM.DURABLE.MODEL.QUEUE. Produkt může výchozí nastavení přepsat zadáním vlastních trvalých a netrvalých modelových front na téma podle jednotlivých témat jako vlastností objektu tématu přidruženého k odběru.

Bez ohledu na atributy zděděné z modelových front nelze znovu použít obslužnou rutinu spravované fronty k vytvoření dalšího odběru. Další manipulační prostředek pro spravovanou frontu nelze získat tak, že otevřete spravovanou frontu podruhé s použitím vráceného názvu fronty. Fronta se chová tak, jako kdyby byla otevřena pro výhradní vstup.

Nespravované fronty jsou flexibilnější než spravované fronty. Můžete například sdílet nespravované fronty, nebo definovat více odběrů v jedné frontě. Další příklad, “Příklad 3: Nespravovaný odběratel MQ” na stránce 282, ukazuje, jak kombinovat odběry s nespravovanou frontou odběrů.

Poznámka: Styl kompaktního kódování je určen pro přehlednost, nikoli pro použití v produkčním prostředí.

Výsledky jsou zobrazeny v Obrázek 46 na stránce 280.

```
#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 44. Odběratel MQ -část 1: deklarace a manipulace s parametry.

V tomto příkladu jsou k dispozici některé další komentáře k deklaracím.

MQHOBJ Hobj = MQHO_NONE;
Produkt nemůže výslovně otevřít frontu netrvalého spravovaného odběru pro příjem publikací, ale je třeba přidělit úložiště pro zpracování objektu, který správce front vrátí při otevření fronty pro produkt. Je důležité inicializovat popisovač na MQHO_OBJECT. Tato hodnota označuje správci front, že je třeba vrátit manipulátor fronty do fronty odběru.

MQSD sd = {MQSD_DEFAULT};
Nový deskriptor odběru použitý v produktu MQSUB.

MQCHAR48 qName;
Although the example doesn't require knowledge of the subscription queue, the example does inquire the name of the subscription queue - the MQINQ binding is a little awkward in the C language, so you might find this part of the example useful to study.

```
    do {
        MQCONN(qmName, &Hconn, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        strncpy(sd.ObjectName, topicName, MQ_TOPIC_NAME_LENGTH);
        sd.ObjectString.VSPtr = topicString;
        sd.ObjectString.VSLength = MQVS_NULL_TERMINATED;
        sd.Options = MQSO_CREATE | MQSO_MANAGED | MQSO_NON_DURABLE | MQSO_FAIL_IF_QUIESCING ;
        sd.ResObjectString.VSPtr = resTopicStr;
        sd.ResObjectString.VSBufSize = sizeof(resTopicStrBuffer)-1;
        MQSUB(Hconn, &sd, &Hobj, &Hsub, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        gmo.Options = MQGMO_WAIT | MQGMO_NO_SYNCPOINT | MQGMO_CONVERT;
        gmo.WaitInterval = 10000;
        inquireQname(Hconn, Hobj, qName);
        printf("Waiting %d seconds for publications matching \"%s\" from \"%-0.48s\"\n",
               gmo.WaitInterval/1000, resTopicStr, qName);
        do {
            memcpy(md.MsgId, MQMI_NONE, sizeof(md.MsgId));
            memcpy(md.CorrelId, MQCI_NONE, sizeof(md.CorrelId));
            md.Encoding       = MQENC_NATIVE;
            md.CodedCharSetId = MQCCSI_Q_MGR;
            memset(publicationBuffer, 0, sizeof(publicationBuffer));
            MQGET(Hconn, Hobj, &md, &gmo, sizeof(publicationBuffer-1),
                publication, &messlen, &CompCode, &Reason);
            if (Reason == MQRC_NONE)
                printf("Received publication \"%s\"\n", publication);
        }
        while (CompCode == MQCC_OK);
        if (CompCode != MQCC_OK && Reason != MQRC_NO_MSG_AVAILABLE) break;
        MQCLOSE(Hconn, &Hsub, MQCO_REMOVE_SUB, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        MQDISC(&Hconn, &CompCode, &Reason);
    } while (0);
    printf("Completion code %d and Return code %d\n", CompCode, Reason);
    return;
}
void inquireQname(MQHCONN Hconn, MQHOBJ Hobj, MQCHAR48 qName) {
#define _selectors 1
#define _intAttrs 1

    MQLONG select[_selectors] = {MQCA_Q_NAME}; /* Array of attribute selectors */
    MQLONG intAttrs[_intAttrs];                /* Array of integer attributes  */
    MQLONG CompCode, Reason;
    MQINQ(Hconn, Hobj, _selectors, select, _intAttrs, intAttrs, MQ_Q_NAME_LENGTH, qName,
          &CompCode, &Reason);
    if (CompCode != MQCC_OK) {
        printf("MQINQ failed with Condition code %d and Reason %d\n", CompCode, Reason);
        strcpy(qName, "unknown queue");
    }
    return;
}
```

Obrázek 45. Odběratel MQ -část 2: kód těla.

```
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 46. Výstup ze spravovaného odběratele produktu MQ

V tomto příkladu jsou k dispozici některé další komentáře k provedení kódu.

strncpy(sd.ObjectName, topicName, MQ_Q_NAME_LENGTH);
Má-li parametr topicName hodnotu null nebo je prázdný *(výchozí hodnota)*, nebude název tématu použit k výpočtu vyřešeného řetězce tématu.

sd.ObjectString.VSPtr = topicString;
Místo použití pouze předdefinovaného objektu tématu v tomto příkladu poskytuje programátor objekt tématu a řetězec tématu, který je zkombinován s použitím produktu MQSUB. Všimněte si, že řetězec tématu má strukturu MQCHARV .

sd.ObjectString.VSLength = MQVS_NULL_TERMINATED;
Alternativa k nastavení délky pole MQCHARV .

sd.Options = MQSO_CREATE | MQSO_MANAGED | MQSO_NON_DURABLE | MQSO_FAIL_IF_QUIESCING;
Po definování řetězce tématu potřebují příznaky produktu sd.Options nejpečlivější pozornost. Existuje mnoho voleb, příklad uvádí pouze ty nejpoužívanější; ostatní jsou ponechány na výchozí .

1. Vzhledem k tomu, že odběr je *netrvalý*, tj. má dobu životnosti otevřeného odběru v aplikaci, nastaví příznak MQSO_CREATE. Pro zlepšení čitelnosti můžete také nastavit příznak *(výchozí)* MQSO_NON_DURABLE .
2. Komplementování MQSO_CREATE je MQSO_RESUME. Oba parametry lze nastavit společně; správce front buď vytvoří nový odběr nebo obnoví existující odběr, podle toho, co je vhodné. Pokud však zadáte MQSO_RESUME , musíte také inicializovat strukturu MQCHARV pro sd.SubName, i když není odběr k obnovení. Selhání inicializace SubName vede k návratovému kódu 2440: MQRC_SUB_NAME_ERROR z MQSUB.

   Poznámka: MQSO_RESUME je vždy ignorován pro netrvalý spravovaný odběr: ale jeho uvedení bez inicializace struktury MQCHARV pro sd.SubName způsobí chybu.
3. Kromě toho existuje třetí příznak, který ovlivňuje způsob, jakým je odběr otevřen, MQSO_ALTER. S ohledem na správná oprávnění se vlastnosti obnovených odběrů mění tak, aby odpovídaly jiným atributům uvedeným v produktu MQSUB.

   Poznámka: Musí být zadán alespoň jeden z parametrů MQSO_CREATE, MQSO_RESUME a MQSO_ALTER . Viz Volby (MQLONG). Existují příklady použití všech tří příznaků v produktu “Příklad 3: Nespravovaný odběratel MQ” na stránce 282.
4. Nastavte produkt MQSO_MANAGED pro správce front tak, aby byl pro vás automaticky určen odběr.

sd.ObjectString.VSLength = MQVS_NULL_TERMINATED;
Volitelně vynechte nastavení délky MQCHARV pro prázdné řetězce s hodnotou null a místo toho použijte příznak ukončení znaku null.

sd.ResObjectString.VSPtr = resTopicStr;
Výsledný řetězec tématu se vypisuje v prvním souboru printf v programu. Nastavte produkt MQCHARV ResObjectString for WebSphere MQ tak, aby vracel přeložený řetězec zpět do programu.

Poznámka: Příkaz resTopicStringBuffer je inicializován na hodnoty null v souboru memset(resTopicStr, 0, sizeof(resTopicStrBuffer)). Vrácené řetězce témat nekončí koncovým znakem null.

sd.ResObjectString.VSBufSize = sizeof(resTopicStrBuffer)-1;
Nastavte velikost vyrovnávací paměti sd.ResObjectString na hodnotu menší, než je její skutečná velikost. Tento zabraňuje přepsání ukončovače null, který je poskytnut, v případě, že vyřešený řetězec tématu vyplní celou vyrovnávací paměť.

Poznámka: Pokud je řetězec tématu delší než sizeof(resTopicStrBuffer)-1, není vrácena žádná chyba. I když VSLength > VSBufSiz délka vrácená v sd.ResObjectString.VSLength je délka celého řetězce a nemusí být nutně délka vráceného řetězce. Testujte sd.ResObjectString.VSLength < sd.ResObjectString.VSBufSiz a potvrďte, že řetězec tématu je dokončen.

MQSUB(Hconn, &sd, &Hobj, &Hsub, &CompCode, &Reason);
Funkce MQSUB vytvoří odběr. Pokud je netrvanlivý, pravděpodobně se o její název nezajímáte, můžete však zkontrolovat jeho stav v Průzkumníku WebSphere MQ . Můžete zadat parametr sd.SubName jako vstup, takže víte, jak se hledat; zřejmě se musíte vyvarovat kolizi názvů s jinými odběry.

MQCLOSE(Hconn, &Hsub, MQCO_REMOVE_SUB, &CompCode, &Reason);
Zavírání odběru a fronty odběru je volitelné. V tomto příkladu je odběr uzavřen, ale ne ve frontě. Volba MQCLOSE MQCO_REMOVE_SUB je standardně v tomto případě, protože odběr je netrvalý. Použití MQCO_KEEP_SUB je chyba.

Poznámka: odběr *fronta* není uzavřen produktem MQSUBa jeho popisovač Hobjzůstává platný do doby, než je fronta uzavřena produktem MQCLOSE nebo MQDISC. Dojde-li k předčasnému ukončení aplikace, dojde k vyčištění fronty a odběru někdy po ukončení aplikace správcem front.

Související pojmy

“Příklad 1: Spotřebitel publikování MQ” na stránce 274
Spotřebitel zpráv produktu MQ je spotřebitel zpráv produktu IBM WebSphere MQ , který se nepřihlašují k odběru vlastních témat.

“Příklad 3: Nespravovaný odběratel MQ” na stránce 282
Nespravovaný odběratel je důležitou třídou aplikace odběratele. S ní kombinujete výhody publikování/odběru s *ovládacím prvkem* řazení do fronty a spotřebou publikací. Tento příklad ukazuje různé způsoby kombinování odběrů a front.

“Zapisování aplikací vydavatele” na stránce 266
Začněte pracovat se zápisem aplikací vydavatele tak, že prostudujete dva příklady. První je modelován co nejpřesněji v okamžiku, kdy aplikace boduje aplikaci do fronty, a druhý demonstruje vytváření témat dynamicky-běžnější vzor pro aplikace vydavatele.

Příklad 3: Nespravovaný odběratel MQ

Nespravovaný odběratel je důležitou třídou aplikace odběratele. S ní kombinujete výhody publikování/odběru s *ovládacím prvkem* řazení do fronty a spotřebou publikací. Tento příklad ukazuje různé způsoby kombinování odběrů a front.

Nespravovaný vzor je častěji přidružen k odběrům *trvalých* než *netrvalých*. Životní cyklus odběru vytvořeného nespravovaným odběratelem je obvykle nezávislý na životním cyklu samotné odběratelské aplikace. Díky tomu, že odběr bude trvalý, obdrží publikování publikování, i když není aktivní žádná odebírající aplikace.

Můžete vytvořit trvalé odběry *spravované* , chcete-li dosáhnout stejného výsledku, ale některé aplikace vyžadují větší flexibilitu a kontrolu nad frontami a zprávami, než je možné u spravovaného odběru. U trvalého spravovaného odběru vytvoří správce front trvalou frontu pro publikování, která se shodují s tématem odběru. Odstraní frontu a přidružená publikování po odstranění odběru.

Obvykle jsou použity trvalé *spravované* odběry, je-li životní cyklus aplikace a odběr v podstatě stejný, ale je těžké zaručit záruku. Díky tomu, že odběr bude trvalý a má-li vydavatel vytvářet trvalé publikování, neexistují žádné ztracené zprávy, pokud by došlo k předčasnému ukončení správce front nebo odběratele a je třeba jej obnovit.

Správce front implicitně otevře trvalou spravovanou frontu odběru pro odběratele takovým způsobem, že sdílené zpracování fronty není možné. Kromě toho nelze pro každou spravovanou frontu vytvořit více než jeden odběr a fronty lze lépe spravovat, protože máte menší kontrolu nad názvy front. Z těchto důvodů zvažte, zda je odběratel *nespravováno* MQ vhodnější pro aplikace vyžadující trvalé odběry než odběratel produktu *spravované* MQ .

Kód v produktu Obrázek 49 na stránce 287 demonstruje nespravovaný vzorek trvalého odběru. Pro ilustraci tento kód také vytváří nespravované, netrvalé odběry. Tento příklad ilustruje následující fazety vzorku:

- Na odběrech poptávky: řetězce témat odběru jsou dynamické. Poskytují ji aplikace, když je spuštěna.
- Zjednodušená správa témat odběru: Správa témat odběru je zjednodušena definováním kořenové části řetězce tématu odběru pomocí administrativně definovaného tématu. Tím skryjete kořenovou část

stromu témat z aplikace. Skrytím kořenové části může být odběratel implementován do různých stromů témat.

- Flexibilní správa odběrů: Můžete definovat odběr buď administrativně, nebo jej vytvořit na vyžádání v programu odběratele. Mezi administrativně a programově vytvořenými odběry neexistuje rozdíl mezi administrativně a programově vytvořenými odběry, s výjimkou atributu, který ukazuje, jak byl vytvořen odběr. Existuje třetí typ odběru, který je vytvořen automaticky správcem front pro distribuci odběrů. Všechny odběry jsou zobrazeny v Průzkumníku WebSphere MQ .
- Flexibilní přidružení odběrů s frontami: Předdefinovaná lokální fronta je přidružena k odběru funkcí MQSUB . Existují různé způsoby použití funkce MQSUB pro přidružení odběrů s frontami:
  - Přidružte odběr ke frontě, která má *ne* existující odběry, `MQSO_CREATE + (Hobj from MQOPEN)`.
  - Přidružte *nový* odběr ke frontě s existujícími odběry `MQSO_CREATE + (Hobj from MQOPEN)`.
  - Přesuňte existující odběr do jiné fronty, `MQSO_ALTER + (Hobj from MQOPEN)`.
  - Obnovte existující odběr přidružený k existující frontě, `MQSO_RESUME + (Hobj = MQHO_NONE)` nebo `MQSO_RESUME + (Hobj = from MQOPEN of queue with existing subscription)` .
  - Kombinací produktu `MQSO_CREATE | MQSO_RESUME | MQSO_ALTER` v různých kombinacích můžete obsloužit různé vstupní stavy odběru a fronty, aniž byste museli kódovat více verzí produktu MQSUB s různými hodnotami `sd.Options` .
  - Případně kódováním specifické volby produktu `MQSO_CREATE | MQSO_RESUME | MQSO_ALTER` správce front vrátí chybu (Tabulka 43 na stránce 284), pokud stavy odběru a fronty poskytnuté jako vstup do MQSUB jsou nekonzistentní s hodnotou `sd.Options`. Obrázek 55 na stránce 290 Zobrazuje výsledky vydání MQSUB pro odběr X s různými individuálními nastaveními parametru `sd.Options` a předáním tří různých manipulátorů objektů.

  Prozkoumejte různé vstupy do ukázkového programu v produktu Obrázek 48 na stránce 286 a seznamte se s těmito různými druhy chyb. Jedna běžná chyba, `RC = 2440`, která není zahrnuta v případech uvedených v tabulce, je chyba názvu odběru. je obvykle způsoben předáním hodnoty null nebo neplatnému názvu odběru s `MQSO_RESUME` nebo `MQSO_ALTER` .
- Multiprocessing: Můžete sdílet mezi mnoha spotřebiteli práci na čtení publikací. Všechny publikace se nacházejí v jedné frontě přidružené k odběru tématu odběru. Spotřebitelé mají možnost volby otevření fronty přímo pomocí produktu `MQOPEN` nebo obnovení odběru pomocí produktu `MQSUB`.
- Spojení odběru: Více odběrů lze vytvořit ve stejné frontě. Buďte opatrní s touto schopností, protože to může vést k "překrývajícím se" odběrům a přijímat stejnou publikaci vícekrát. Volba `MQSO_GROUP_SUB` eliminuje duplicitní publikování v důsledku překrývajících se odběrů.
- Subscriber and consumer separation: stejně jako tři modely zákazníků ilustrované v příkladech je dalším modelem oddělit odběratele od odběratele. Jedná se o variantu nespravovaného odběratele MQ Subscriber, ale spíše než vydání `MQOPEN` a `MQSUB` ve stejném programu, jeden program se přihlásí k odběru publikací a jiný program je spotřebovává. Odběratel může být například součástí klastru publikování/odběru a konzumenta připojený ke správci front mimo klastr správců front. Konzument přijímá publikace prostřednictvím standardního distribuovaného řazení do fronty tím, že definuje frontu odběru jako definici vzdálené fronty.

Porozumění chování produktu `MQSO_CREATE | MQSO_RESUME | MQSO_ALTER` je důležité, zvláště pokud plánujete zjednodušit svůj kód pomocí kombinací těchto voleb. Prostudujte tabulku Tabulka 43 na stránce 284 , která zobrazuje výsledky předávání různých obslužných rutin front do fronty MQSUB, a výsledky spuštění ukázkového programu uvedeného v tématu Obrázek 50 na stránce 288 na produkt Obrázek 55 na stránce 290.

Scénář použitý ke konstrukci tabulky má jeden odběr X a dvě fronty, A a B . Parametr názvu odběru `sd.SubName` je nastaven na hodnotu X, název odběru připojený ke frontě A. K frontě B není připojen žádný odběr.

V produktu Tabulka 43 na stránce 284je MQSUB předán odběr X a popisovač fronty do fronty A. Výsledky voleb odběru jsou následující:

- MQSO_CREATE selže, protože manipulační prostředek fronty odpovídá frontě A , která již má odběr produktu X. Porovnejte toto chování s úspěšným voláním. Toto volání je úspěšné, protože fronta B nemá k sobě přiložený odběr X .
- MQSO_RESUME je úspěšný, protože manipulátor fronty odpovídá frontě A , která již má odběr produktu X. Naopak volání selže, pokud odběr X ve frontě Aneexistuje.
- Produkt MQSO_ALTER se chová podobným způsobem jako MQSO_RESUME s ohledem na otevření odběru a fronty. Pokud však atributy obsažené v deskriptoru odběru předané produktu MQSUB se liší od atributů odběru, MQSO_RESUME se nezdaří, zatímco MQSO_ALTER uspěje, pokud má instance programu oprávnění k pozměnění atributů. Všimněte si, že řetězec tématu nelze nikdy změnit v odběru, ale spíše než vrátit chybu, produkt MQSUB ignoruje hodnoty názvu tématu a řetězce tématu v deskriptoru odběru a použije hodnoty v existujícím odběru.

Dále se podívejte na Tabulka 43 na stránce 284 , kde MQSUB prošel odběrem X a popisovač fronty do fronty B. Výsledky voleb odběru jsou následující:

- Produkt MQSO_CREATE uspěje a vytvoří odběr X ve frontě B , protože se jedná o nový odběr ve frontě B.
- MQSO_RESUME selže. Příkaz MQSUB hledá odběr X ve frontě B a nenalezne jej, ale nevrací *RC = 2428-odběr X neexistuje* , vrací *RC = 2019-Fronta odběru neodpovídá manipulátoru objektu fronty*. Chování třetí volby MQSO_ALTER napovídá o příčině této neočekávané chyby. MQSUB očekává, že se popisovač fronty bude odkazovat na frontu s odběrem. Před kontrolou, zda existuje odběr uvedený v produktu sd . SubName , je nejprve tato kontrola provedena.
- MQSO_ALTER uspěje a přesune odběr z fronty A do fronty B.

Případ, který není zobrazen v tabulce, je případ, kdy název odběru odběru ve frontě A neodpovídá názvu odběru v produktu sd . SubName. Toto volání selhává s *RC = 2428-odběr X neexistuje ve frontě A* .

Tabulka 43. Chyby z MQSUB s různými manipulátory front a kombinace odběrů

	Fronta A **Odběr X** **Fronta B** **Bez odběru**	**Fronta A** **Bez odběru** **Fronta B** **Bez odběru**
Objekt Hobj pro frontu *Queue A* předaný funkci MQSUB	**MQSO_CREATE** RC = 2432-Odběr X již ve frontě A existuje **MQSO_RESUME** Obnoví odběr X ve frontě A **MQSO_ALTER** Obnoví odběr X ve frontě A a povolí změny	**MQSO_CREATE** Vytvoří odběr X ve frontě A **MQSO_RESUME** RC = 2428-Odběr X ve frontě A neexistuje **MQSO_ALTER** RC = 2428-Odběr X ve frontě A neexistuje
Objekt Hobj pro frontu *Queue B* předaný funkci MQSUB	**MQSO_CREATE** Vytvoří nový odběr X ve frontě B **MQSO_RESUME** RC = 2019-Fronta odběru se neshoduje s manipulátorem objektu fronty **MQSO_ALTER** Přesunout odběr X z fronty A do fronty B	**MQSO_CREATE** Vytvoří nový odběr X ve frontě B **MQSO_RESUME** RC = 2428-odběr X na frontě B neexistuje **MQSO_ALTER** RC = 2428-odběr X na frontě B neexistuje

Tabulka 43. Chyby z MQSUB s různými manipulátory front a kombinace odběrů (pokračování)		
	Fronta A **Odběr X** **Fronta B** **Bez odběru**	**Fronta A** **Bez odběru** **Fronta B** **Bez odběru**
MQHO_NONE předán rutině MQSUB	**MQSO_CREATE** RC = 2019-Špatný popisovač objektu: nastaví příznak MQSO_MANAGED pro vytvoření spravovaného odběru a vytvoření spravované fronty **MQSO_RESUME** Obnoví odběr X ve frontě A a vrátí objekt Hobj do fronty A **MQSO_ALTER** Obnoví odběr X ve frontě A, vrátí objekt Hobj do fronty A a povolí změny.	**MQSO_CREATE** RC = 2019-Špatný popisovač objektu: nastaví příznak MQSO_MANAGED pro vytvoření spravovaného odběru a vytvoření spravované fronty **MQSO_RESUME** RC = 2428-Žádné předplatné X **MQSO_ALTER** RC = 2019-Špatný popisovač objektu: Žádná fronta A nebo B

Poznámka: Styl kompaktního kódování je určen pro přehlednost, nikoli pro použití v produkčním prostředí.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <cmqc.h>

void inquireQname(MQHCONN HConn, MQHOBJ Hobj, MQCHAR48 qName);

int main(int argc, char **argv)
{
    MQCHAR48 topicNameDefault            = "STOCKS";
    char     topicStringDefault[]        = "IBM/PRICE";
    char     subscriptionNameDefault[]   = "IBMSTOCKPRICESUB";
    char     subscriptionQueueDefault[]  = "STOCKTICKER";
    char     publicationBuffer[101];      /* Allocate to receive messages     */
    char     resTopicStrBuffer[151];      /* Allocate to resolve topic string */
    MQCHAR48 qmName = "";                 /* Default queue manager            */
    MQCHAR48 qName = "";                  /* Allocate storage for MQINQ       */

    MQHCONN  Hconn = MQHC_UNUSABLE_HCONN; /* connection handle                */
    MQHOBJ   Hobj = MQHO_NONE;            /* subscription queue handle        */
    MQHOBJ   Hsub = MQSO_NONE;            /* subscription handle              */
    MQLONG   CompCode = MQCC_OK;          /* completion code                  */
    MQLONG   Reason = MQRC_NONE;          /* reason code                      */
    MQLONG   messlen = 0;
    MQOD     od = {MQOD_DEFAULT};         /* Unmanaged subscription queue     */
    MQSD     sd = {MQSD_DEFAULT};         /* Subscription Descriptor          */
    MQMD     md = {MQMD_DEFAULT};         /* Message Descriptor               */
    MQGMO   gmo = {MQGMO_DEFAULT};        /* get message options              */
    MQLONG   sdOptions = MQSO_CREATE | MQSO_RESUME | MQSO_DURABLE |
MQSO_FAIL_IF_QUIESCING;

    char *   topicName = topicNameDefault;
    char *   topicString = topicStringDefault;
    char *   subscriptionName =  subscriptionNameDefault;
    char *   subscriptionQueue = subscriptionQueueDefault;
    char *   publication = publicationBuffer;
    char *   resTopicStr = resTopicStrBuffer;
    memset(resTopicStrBuffer, 0, sizeof(resTopicStrBuffer));
```

Obrázek 47. 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 48. Nespravovaný odběratel MQ -část 2: manipulace s parametry.

Další komentáře týkající se zacházení s parametry v tomto příkladu jsou následující:

switch((argv[5][0]))
Máte možnost volby Alter | C reate | Resume v parametru 5, abyste otestovali účinek přepisující části nastavení parametru MQSUB , který je v příkladu použit jako výchozí. Výchozí nastavení použité v příkladu je MQSO_CREATE | MQSO_RESUME | MQSO_DURABLE.

Poznámka: Nastavení MQSO_ALTER nebo MQSO_RESUME bez nastavení MQSO_DURABLE je chyba a sd.SubName musí být nastaven a odkazovat na odběr, který může být obnoven nebo změněn.

***subscriptionQueue = '\0';**
sdOptions = sdOptions + MQSO_MANAGED;
Je-li výchozí fronta odběru, STOCKTICKER je nahrazena řetězcem s hodnotou null, pokud je nastaven parametr MQSO_CREATE , tento příklad nastaví příznak MQSO_MANAGED a vytvoří frontu dynamického odběru. Je-li Alter or Resume nastaveno v pátém parametru, chování příkladu bude záviset na hodnotě subscriptionName.

***subscriptionName = '\0';**
sdOptions = sdOptions - MQSO_DURABLE;
Je-li standardní odběr IBMSTOCKPRICESUB nahrazen řetězcem s hodnotou null, odebere tento příkaz příznak MQSO_DURABLE . Spustíte-li příklad poskytující výchozí hodnoty pro ostatní parametry, vytvoří se další dočasný odběr určený pro STOCKTICKER a přijme duplicitní publikace. Při příštím spuštění tohoto příkladu, bez parametrů, obdržíte znovu pouze jednu publikaci.

```
    do {
        MQCONN(qmName, &Hconn, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        if (strlen(subscriptionQueue)) {
            strncpy(od.ObjectName, subscriptionQueue, MQ_Q_NAME_LENGTH);
            MQOPEN(Hconn, &od, MQOO_INPUT_AS_Q_DEF | MQOO_FAIL_IF_QUIESCING | MQOO_INQUIRE,
                  &Hobj, &CompCode, &Reason);
            if (CompCode != MQCC_OK) break;
        }
        strncpy(sd.ObjectName, topicName, MQ_TOPIC_NAME_LENGTH);
        sd.ObjectString.VSPtr = topicString;
        sd.ObjectString.VSLength = MQVS_NULL_TERMINATED;
        sd.SubName.VSPtr = subscriptionName;
        sd.SubName.VSLength = MQVS_NULL_TERMINATED;
        sd.ResObjectString.VSPtr = resTopicStr;
        sd.ResObjectString.VSBufSize = sizeof(resTopicStrBuffer)-1;
        MQSUB(Hconn, &sd, &Hobj, &Hsub, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        gmo.Options = MQGMO_WAIT | MQGMO_NO_SYNCPOINT | MQGMO_CONVERT;
        gmo.WaitInterval = 10000;
        gmo.MatchOptions = MQMO_MATCH_CORREL_ID;
        memcpy(md.CorrelId, sd.SubCorrelId, MQ_CORREL_ID_LENGTH);
        inquireQname(Hconn, Hobj, qName);
        printf("Waiting %d seconds for publications matching \"%s\" from %-0.48s\n",
               gmo.WaitInterval/1000, resTopicStr, qName);
        do {
            memcpy(md.MsgId, MQMI_NONE, sizeof(md.MsgId));
            memcpy(md.CorrelId, MQCI_NONE, sizeof(md.CorrelId));
            md.Encoding       = MQENC_NATIVE;
            md.CodedCharSetId = MQCCSI_Q_MGR;
            MQGET(Hconn, Hobj, &md, &gmo, sizeof(publication), publication, &messlen,
&CompCode, &Reason);
            if (Reason == MQRC_NONE)
                printf("Received publication \"%s\"\n", publication);
        }
        while (CompCode == MQCC_OK);
        if (CompCode != MQCC_OK && Reason != MQRC_NO_MSG_AVAILABLE) break;
        MQCLOSE(Hconn, &Hsub, MQCO_NONE, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        MQCLOSE(Hconn, &Hobj, MQCO_NONE, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        MQDISC(&Hconn, &CompCode, &Reason);
    } while (0);
    printf("Completion code %d and Return code %d\n", CompCode, Reason);
}
void inquireQname(MQHCONN Hconn, MQHOBJ Hobj, MQCHAR48 qName) {
#define _selectors 1
#define _intAttrs 1

    MQLONG select[_selectors] = {MQCA_Q_NAME}; /* Array of attribute selectors    */
    MQLONG intAttrs[_intAttrs];            /* Array of integer attributes           */
    MQLONG CompCode, Reason;
    MQINQ(Hconn, Hobj, _selectors, select, _intAttrs, intAttrs, MQ_Q_NAME_LENGTH, qName,
&CompCode, &Reason);
    if (CompCode != MQCC_OK) {
        printf("MQINQ failed with Condition code %d and Reason %d\n", CompCode, Reason);
        strncpy(qName, "unknown queue", MQ_Q_NAME_LENGTH);
    }
    return;
}
```

Obrázek 49. Nespravovaný odběratel MQ -část 3: tělo kódu.

Další komentáře k kódu v tomto příkladu jsou následující:

if (strlen(subscriptionQueue))
Pokud neexistuje žádný název fronty odběru, pak příklad používá MQHO_NONE jako hodnotu Hobj.

MQOPEN(...);
Je otevřena fronta odběrů a popisovač fronty uložen v produktu Hobj.

MQSUB(Hconn, &sd, &Hobj, &Hsub, &CompCode, &Reason);
Odběr se otevře pomocí produktu Hobj , který prošel produktem MQOPEN (nebo MQHO_NONE , pokud neexistuje žádný název fronty odběru). Nespravovaná fronta může být obnovena bez explicitního otevření s použitím MQOPEN.

MQCLOSE(Hconn, &Hsub, MQCO_NONE, &CompCode, &Reason);
Odběr je uzavřen pomocí popisovače odběru. V závislosti na tom, zda je odběr trvalý či nikoli, je odběr uzavřen s implicitním produktem MQCO_KEEP_SUB nebo MQCO_REMOVE_SUB. Trvalý odběr můžete uzavřít s produktem MQCO_REMOVE_SUB, ale *nelze* zavřít netrvalý odběr s produktem MQCO_KEEP_SUB. Akce produktu MQCO_REMOVE_SUB odebere odběr, který zastaví jakékoli další publikace odesílané do fronty odběru.

MQCLOSE(Hconn, &Hobj, MQCO_NONE, &CompCode, &Reason);
Pokud je odběr nespravovaný, není provedena žádná speciální akce. Je-li fronta spravována a odběr je uzavřen explicitním nebo implicitním produktem MQCO_REMOVE_SUB, budou všechny publikace vymazány z fronty a fronty odstraněné v tomto bodu.

gmo.MatchOptions = MQMO_MATCH_CORREL_ID;
memcpy(md.CorrelId, sd.SubCorrelId, MQ_CORREL_ID_LENGTH);
Ujistěte se, že přijaté zprávy jsou ty, které jsou pro náš odběr.

Výsledky z příkladu ilustrují aspekty publikování/odběru:

V produktu Obrázek 50 na stránce 288 se příklad spustí publikováním 130 na téma NYSE/IBM/PRICE .

```
W:\Subscribe3\Debug>..\..\Publish2\Debug\publishstock
Provide parameters:  TopicObject TopicString Publication
Publish "130" to topic "STOCKS" and topic string "IBM/PRICE"
Published "130" to topic string "NYSE/IBM/PRICE"
Completion code 0 and Return code 0
```

Obrázek 50. Publikovat 130 na NYSE/IBM/PRICE

V Obrázek 51 na stránce 288 provedení příkladu s použitím výchozích parametrů přijímá zachované publikování 130. Zadaný objekt tématu a řetězec tématu jsou ignorovány, jak je zobrazeno v části Obrázek 55 na stránce 290. Objekt tématu a řetězec tématu jsou vždy převzaty z objektu odběru, je-li zadán, a řetězec tématu je neměnný. Skutečné chování tohoto příkladu závisí na volbě nebo kombinaci MQSO_CREATE, MQSO_RESUMEa MQSO_ALTER . V tomto příkladě je vybrána volba MQSO_RESUME .

```
W:\Subscribe3\Debug>solution3
Optional parameters: topicName, topicString, subscriptionName, subscriptionQueue, A(lter)|
C(reate)|R(esume)
Values "STOCKS"  "IBM/PRICE" "IBMSTOCKPRICESUB"  "STOCKTICKER" sd.Options=8206
Waiting 10 seconds for publications matching "NYSE/IBM/PRICE" from STOCKTICKER
Received publication "130"
Completion code 0 and Return code 0
```

Obrázek 51. Přijmout zachované publikování

V (Obrázek 52 na stránce 289) nejsou přijata žádná publikování, protože trvanlivý odběr již obdržel zachované publikování. V tomto příkladu bude odběr obnoven tak, že zadáte pouze název odběru bez názvu fronty. Pokud byl zadán název fronty, byla by nejprve otevřena fronta a obsluha byla předána produktu MQSUB.

Poznámka: Chyba 2038 z MQINQ je způsobena implicitní MQOPEN z STOCKTICKER tím, že MQSUB nezahrnuje volbu MQOO_INQUIRE . Vyvarovat se návratového kódu 2038 z MQINQ tím, že otevřete frontu explicitně.

```
W:\Subscribe3\Debug>solution3 STOCKS IBM/PRICE IBMSTOCKPRICESUB / Resume
Optional parameters: topicName, topicString, subscriptionName, subscriptionQueue, A(lter)|
C(reate)|R(esume)
Values "STOCKS"  "IBM/PRICE" "IBMSTOCKPRICESUB"  "" sd.Options=8204
MQINQ failed with Condition code 2 and Reason 2038
Waiting 10 seconds for publications matching "NYSE/IBM/PRICE" from unknown queue
Completion code 0 and Return code 0
```

Obrázek 52. Obnovit odběr

V produktu Obrázek 53 na stránce 289 vytváří příklad netrvalý nespravovaný odběr pomocí parametru STOCKTICKER jako místo určení. Vzhledem k tomu, že se jedná o nový odběr, obdrží zachované publikování.

```
W:\Subscribe3\Debug>solution3 STOCKS IBM/PRICE / STOCKTICKER Create
Optional parameters: topicName, topicString, subscriptionName, subscriptionQueue, A(lter)|
C(reate)|R(esume)
Values "STOCKS"  "IBM/PRICE" ""  "STOCKTICKER" sd.Options=8194
Waiting 10 seconds for publications matching "NYSE/IBM/PRICE" from STOCKTICKER
Received publication "130"
Completion code 0 and Return code 0
```

Obrázek 53. Přijmout zachované publikování s novým nespravovaným nestálým odběrem

Chcete-li demonstrovat překrývající se odběry v produktu Obrázek 54 na stránce 289 , je odeslána jiná publikace, která mění zachované publikování. Dále je vytvořen nový netrvalý nespravovaný odběr, který neposkytuje název odběru. Zachované publikování je přijato dvakrát, jednou pro nový odběr, a jednou pro trvalý odběr IBMSTOCKPRICESUB , který je stále aktivní ve frontě STOCKTICKER . Příklad je obrázek, že se jedná o frontu, která má odběry, nikoli aplikaci. I když neodkazujete na odběr IBMSTOCKPRICESUB v tomto vyvolání aplikace, aplikace obdrží tuto publikaci dvakrát: jednou z trvalého odběru, který byl vytvořen administrativně, a jednou z netrvalého odběru vytvořeného aplikací samotnou.

```
W:\Subscribe3\Debug>..\..\Publish2\Debug\publishstock
Provide parameters:  TopicObject TopicString Publication
Publish "130" to topic "STOCKS" and topic string "IBM/PRICE"
Published "130" to topic string "NYSE/IBM/PRICE"
Completion code 0 and Return code 0

W:\Subscribe3\Debug>solution3 STOCKS IBM/PRICE / STOCKTICKER Create
Optional parameters: topicName, topicString, subscriptionName, subscriptionQueue, A(lter)|
C(reate)|R(esume)
Values "STOCKS"  "IBM/PRICE" ""  "STOCKTICKER" sd.Options=8194
Waiting 10 seconds for publications matching "NYSE/IBM/PRICE" from STOCKTICKER
Received publication "130"
Received publication "130"
Completion code 0 and Return code 0
```

Obrázek 54. Překrývání odběrů

V produktu Obrázek 55 na stránce 290 tento příklad ukazuje, že zadání nového řetězce tématu a existujícího odběru nevedlo ke změně odběru.

1. V prvním případě produkt Resume obnoví existující odběr, jak byste mohli očekávat, a ignoruje změněný řetězec tématu.
2. Ve druhém případě, Alter způsobí chybu, RC = 2510, Topic not alterable.
3. Ve třetím příkladu příkaz Create způsobí chybu RC = 2432, Sub already exists.

```
W:\Subscribe3\Debug>solution3 "" NASDAC/IBM/PRICE IBMSTOCKPRICESUB STOCKTICKER Resume
Optional parameters: topicName, topicString, subscriptionName, subscriptionQueue, A(lter)|C(reate)|R(esume)
Values ""  "NASDAC/IBM/PRICE" "IBMSTOCKPRICESUB"  "STOCKTICKER" sd.Options=8204
Waiting 10 seconds for publications matching "NYSE/IBM/PRICE" from STOCKTICKER
Received publication "130"
Completion code 0 and Return code 0

W:\Subscribe3\Debug>solution3 "" NASDAC/IBM/PRICE IBMSTOCKPRICESUB STOCKTICKER Alter
Optional parameters: topicName, topicString, subscriptionName, subscriptionQueue, A(lter)|C(reate)|R(esume)
Values ""  "NASDAC/IBM/PRICE" "IBMSTOCKPRICESUB"  "STOCKTICKER" sd.Options=8201
Completion code 2 and Return code 2510

W:\Subscribe3\Debug>solution3 "" NASDAC/IBM/PRICE IBMSTOCKPRICESUB STOCKTICKER Create
Optional parameters: topicName, topicString, subscriptionName, subscriptionQueue, A(lter)|C(reate)|R(esume)
Values ""  "NASDAC/IBM/PRICE" "IBMSTOCKPRICESUB"  "STOCKTICKER" sd.Options=8202
Completion code 2 and Return code 2432
```

Obrázek 55. Témata odběru nelze změnit

Související pojmy

“Příklad 1: Spotřebitel publikování MQ” na stránce 274
Spotřebitel zpráv produktu MQ je spotřebitel zpráv produktu IBM WebSphere MQ , který se nepřihlašují k odběru vlastních témat.

“Příklad 2: Spravovaný odběratel produktu MQ” na stránce 277
Odběratel MQ je upřednostňovaným vzorem pro většinu aplikací odběratele. Příklad vyžaduje *ne* administrative definition of queues, témata nebo .

“Zapisování aplikací vydavatele” na stránce 266
Začněte pracovat se zápisem aplikací vydavatele tak, že prostudujete dva příklady. První je modelován co nejpřesněji v okamžiku, kdy aplikace boduje aplikaci do fronty, a druhý demonstruje vytváření témat dynamicky-běžnější vzor pro aplikace vydavatele.

Životní cykly publikování/odběru

Zvažte životní cykly témat, odběrů, odběratelů, publikací, vydavatelů a front v aplikacích pro návrh aplikací publikování/odběru.

Životní cyklus objektu, jako např. odběr, začíná jeho vytvořením a končí jeho odstraněním. Může také obsahovat jiné stavy a změny, které prochází, jako je dočasné pozastavení, které má nadřízená a podřízená témata, vypršení platnosti a odstranění.

Tradičně jsou objekty produktu WebSphere MQ , jako jsou fronty, vytvořeny administrativně nebo prostřednictvím administračních programů s použitím formátu PCF (Programmable Command Format). Publikování/odběr se liší v poskytování příkazových slov MQSUB a MQCLOSE k vytváření a odstraňování odběrů, které mají koncept spravovaných odběrů, které nejen vytvářejí a odstraňují fronty, ale také čistí nespotřebované zprávy a mají přidružení mezi administrativně vytvořenými objekty tématu a programově nebo administrativně vytvořeným řetězem témat.

Tento funkční bohatost zajišťuje širokou škálu požadavků na publikování/odběr a také zjednodušuje návrh některých běžných vzorců aplikace publikování/odběru. Spravované odběry například zjednodušují programování i administraci odběru, který je určen pouze tak dlouho jako program, který jej vytvořil. Nespravované odběry zjednodušují programování tam, kde dochází k uvolnění připojení mezi odebírajícími a odběratelskými publikacemi. Centrálně vytvořené odběry jsou užitečné v případech, kdy je vzorek jedním ze směrování přenosu publikování na spotřebitele na základě centralizovaného modelu řízení, například odesílání informací o letu do automatizovaných bran, zatímco programově vytvořené odběry mohou být použity, jsou-li zaměstnanci odpovědní za přihlášení k tomuto letu odpovědní za přihlášení k odběru pro daný let, zadáním čísla letu na bránu.

V tomto posledním příkladu může být spravovaný trvalý odběr vhodný: spravovaný, protože odběry jsou vytvářeny velmi často a mají jasný koncový bod při zavření brány a odběr může být programově odebrán; trvalý, aby nedošlo ke ztrátě záznamu o cestujících kvůli tomu, že program odběratele brány má nějaký důvod nebo jiný důvod.[2]. Chcete-li zahájit zveřejnění záznamů cestujících k bráně, možný návrh by byl pro použití brány pro obě přihlášení k odběru na osobní záznamy pomocí čísla brány, a publikovat události otevření brány s použitím čísla brány. Vydavatel odpovídá na událost otevření brány publikováním

[2] Vydavatel musí odeslat osobní záznamy jako trvalé zprávy, aby se zabránilo dalším možným poruchám, samozřejmě.

záznamů o cestujících-což by pak mohlo jít také do dalších zainteresovaných stran, jako je vyúčtování, na záznam letu, a na služby zákazníkům, na textová upozornění na mobilní telefony cestujících k bráně číslo.

Centrálně spravovaný odběr může použít trvalý nespravovaný model, směrovat seznamy cestujících k bráně pomocí předem definované fronty pro každou bránu.

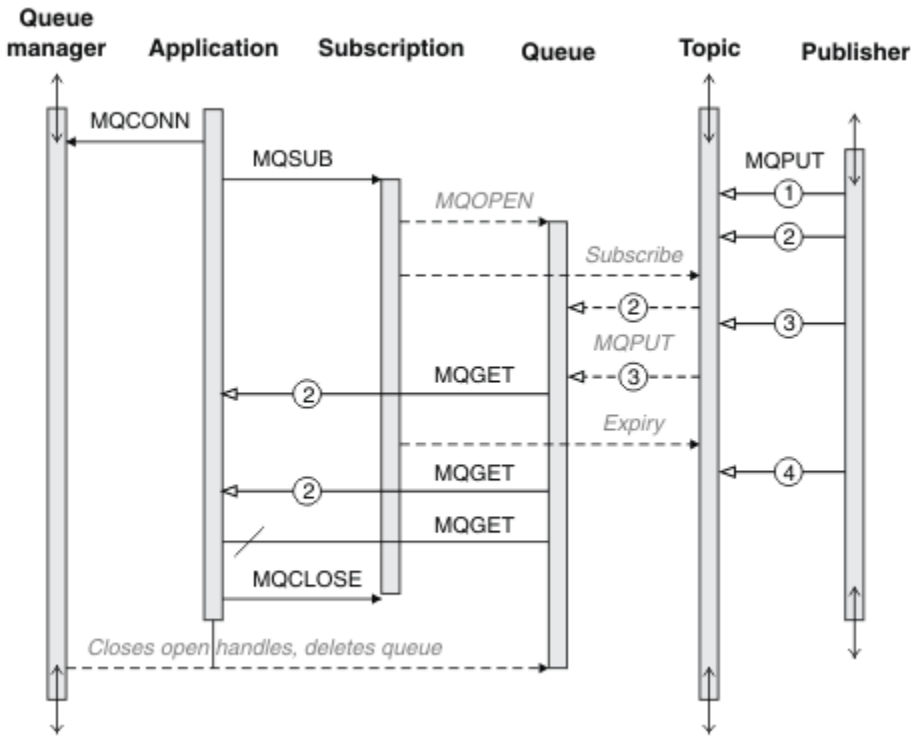
Následující tři příklady životních cyklů publikování/odběru ilustrují způsob interakce spravovaného netrvalého, spravovaného trvalého a nespravovaného trvalého odběratele s odběry, tématy, frontami, vydavateli a správcem front a jak mohou být odpovědnosti rozděleny mezi administrací a programy odběratele.

Spravovaný netrvalý odběratel

Produkt Obrázek 56 na stránce 291 zobrazuje aplikaci vytvářející spravovaný netrvalý odběr, získávání dvou zpráv, které jsou publikovány v rámci tématu identifikovaného v odběru a které se ukončuje. Interakce označené šedým kurzívou s tečkovými šipkami jsou implicitní.

Je třeba poznamenat, že existují určité body.

1. Aplikace vytvoří odběr na téma, které již bylo publikováno dvakrát. Když odběratel obdrží svou první publikaci, obdrží *druhé* publikování, které je aktuálně zachovaným publikováním.
2. Správce front vytvoří dočasnou frontu odběru a vytvoří odběr pro dané téma.
3. Odběr má vypršení platnosti. Po vypršení platnosti odběru nejsou k tomuto odběru odeslány žádné další publikace, ale odběratel bude nadále dostávat zprávy publikované před tím, než vyprší platnost odběru. Vypršení platnosti publikování není ovlivněno vypršením platnosti odběru.
4. Čtvrtá publikace není umístěna ve frontě odběru a v důsledku toho poslední MQGET tuto publikaci nevrací.
5. Přestože odběratel zavře svůj odběr, nezavře své připojení k frontě nebo správci front.
6. Správce front se vyčistí krátce po ukončení aplikace. Vzhledem k tomu, že odběr je spravovaný a netrvalý, je odstraněna fronta odběru.



Obrázek 56. Spravované netrvalé odběratele lifelines

Spravovaný trvalý odběratel

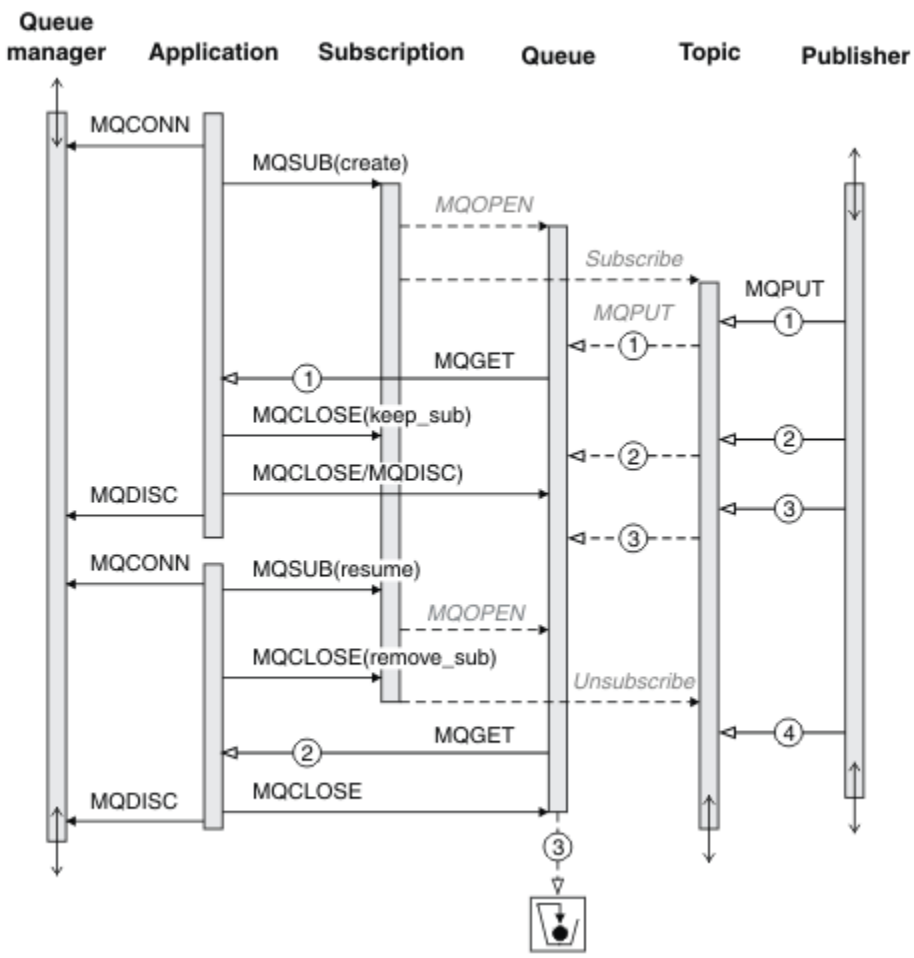
Spravovaný trvalý odběratel převezme předchozí krok dále a zobrazí spravovaný odběr, který přežívuje ukončení a restartování odběratelské aplikace.

Je zde několik nových bodů k poznámce.

1. V tomto příkladu, na rozdíl od posledního, téma publikování neexistovalo dříve, než bylo definováno v odběru.
2. Když se odběratel ukončí poprvé, zavře odběr s volbou MQCO_KEEP_SUB. To je výchozí chování pro implicitní zavření spravovaného trvalého odběru.
3. Když odběratel obnoví odběr, je znovu otevřena fronta odběrů.
4. Nová publikace 2, umístěná ve frontě před jejím opětným otevřením, je k dispozici pro MQGETi po odebrání odběru.

   Přestože je odběr trvalý, odběratel spolehlivě přijímá všechny zprávy odeslané vydavatelem pouze v případě, že *obojí* je trvalý a trvalý. Perzistence zpráv závisí na nastavení pole Persistent v souboru MQMD zprávy odeslaného vydavatelem. Odběratel nemá nad tím žádnou kontrolu.
5. Zavřením odběru s příznakem MQCO_REMOVE_SUB dojde k odebrání odběru a zastavování všech dalších publikování, která jsou umístěna ve frontě odběru. Po zavření fronty odběru odebere správce front nepřečtenou publikaci 3a poté ji odstraní. Akce je ekvivalentní administrativnímu odstranění odběru.

   Poznámka: Neodstraňujte frontu ručně, nebo zadejte příkaz MQCLOSE s volbou MQCO_DELETEnebo MQCO_PURGE_DELETE. Viditelné podrobnosti implementace spravovaného odběru nejsou součástí podporovaného rozhraní produktu WebSphere MQ . Správa správce front nemůže spolehlivě spravovat odběr, pokud nemá úplnou kontrolu.



Obrázek 57. Lifeliny spravovaného trvalého odběratele

Nespravovaný trvalý odběratel

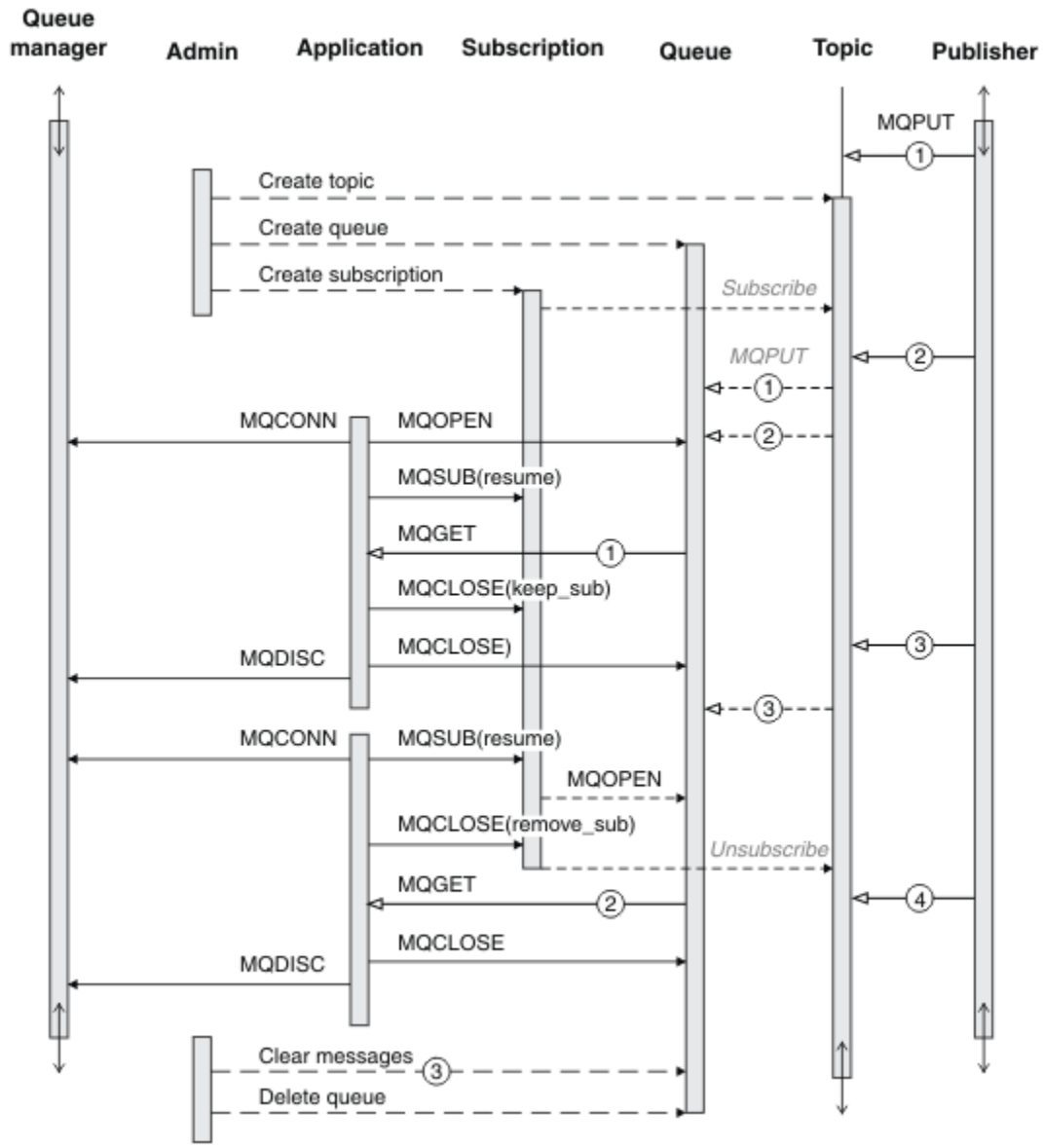
Administrátor je přidán do třetího příkladu: nespravovaný trvalý odběratel. Je to dobrý příklad, který ukazuje, jak může administrátor komunikovat s aplikací publikování/odběru.

Zobrazí se body, které se mají zobrazit.

1. Vydavatel umístí zprávu 1na téma, které později bude přidruženo k objektu tématu, který se používá pro odběr. Objekt tématu definuje řetězec tématu, který se shoduje s tématem, které bylo publikováno pomocí zástupných znaků.
2. Téma má zachované publikování.
3. Administrátor vytvoří objekt tématu, frontu a odběr. Objekt tématu a fronta musí být definovány před odběrem.
4. Aplikace otevře frontu přidruženou k odběru a předá `MQSUB` manipulátoru fronty. Může to alternativně otevřít odběr a předat mu popisovač fronty `MQHO_NONE`. Příkaz converse není pravdivý, nelze obnovit odběr tím, že mu projde pouze popisovač fronty bez názvu odběru-fronta může mít více odběrů.
5. Aplikace otevře odběr s použitím volby `MQSO_RESUME` , i když je prvním otevření odběru poprvé. Obnovuje se administrativně vytvořený odběr.
6. Odběratel obdrží zachované publikování 1. Publikování 2, ačkoli bylo publikováno před přijetím jakýchkoli publikování odběratelem, bylo publikováno po spuštění odběru a je druhé publikování ve frontě odběru.

Poznámka: Pokud zachované publikování není publikováno jako trvalá zpráva, je po restartování správce front ztraceno.

7. V tomto příkladu je odběr trvalý. Je možné, aby program vytvořil nespravovaný netrvalý odběr. Mělo by být zřejmé, že to není něco, co by měl administrátor dělat.
8. Efekt volby MQCO_REMOVE_SUB při zavření odběru je odebrání odběru, jako kdyby jej administrátor odstranil. Tím se zastaví jakékoli další publikace odesílané do fronty, ale neovlivní publikování, která jsou již ve frontě, i když je fronta uzavřena, na rozdíl od *spravovaného* trvalého odběru.
9. Administrátor později odstraní zbývající zprávu, 3a odstraní frontu.



Obrázek 58. Nespravované linie lifeline

Normální vzor pro nespravovaný odběr je určen pro úklid front a odběrů, který má provádět administrátor. Obvykle se nikdo nepokusí emulovat chování spravovaného odběratele a uklidit fronty a odběry programově v kódu aplikace. Pokud zjistíte, že potřebujete zapsat logiku správy, otázka, zda můžete dosáhnout stejných výsledků pomocí spravovaného vzorku. Není jednoduché napsat úzce synchronizovaný, zcela spolehlivý kód managementu. Je jednodušší uklidit později, buď ručně, nebo pomocí automatizovaného programu správy, když si můžete být jisti, že zprávy, odběry a fronty mohou být jednoduše vymazány, bez ohledu na jejich stav.

Vlastnosti zprávy publikování/odběru

Produkt WebSphere MQ publish/subscribe messaging se týká více vlastností zprávy.

Token PubAccounting

Jedná se o hodnotu, která bude uvedena v poli AccountingToken deskriptoru zpráv (MQMD) všech publikovaných zpráv, které odpovídají tomuto odběru. AccountingToken je součástí kontextu identity zprávy. Další informace o kontextu zprávy viz “kontext zprávy” na stránce 37. Další informace o poli AccountingToken v produktu MQMD najdete v tématu AccountingToken.

PubApplIdentityData

Jedná se o hodnotu, která bude v poli Data ApplIdentitydeskriptoru zpráv (MQMD) všech publikovaných zpráv, odpovídajících tomuto odběru. ApplIdentityData jsou součástí kontextu identity zprávy. Další informace o kontextu zprávy viz “kontext zprávy” na stránce 37. Další informace o datovém poli ApplIdentityv produktu MQMD najdete v tématu Data aplikaceApplIdentity.

Není-li určena volba MQSO_SET_IDENTITY_CONTEXT, budou data ApplIdentity, která budou nastavena v každé zprávě publikované pro tento odběr, prázdná, jako výchozí kontextové informace.

Je-li určena volba MQSO_SET_IDENTITY_CONTEXT, generuje se uživatel PubApplIdentityData a toto pole je vstupní pole, které obsahuje data produktu ApplIdentity, která mají být nastavena v každé publikaci pro tento odběr.

PubPriority

Jedná se o hodnotu, která bude v poli Priorita deskriptoru zpráv (MQMD) všech publikovaných zpráv, odpovídajících tomuto odběru. Další informace o poli Priorita v MQMD naleznete v tématu Priorita.

Hodnota musí být větší než nula nebo rovna nule; nula je nejnižší priorita. Mohou být použity také následující speciální hodnoty:

- MQPRI_PRIORITY_AS_Q_DEF-Je-li fronta odběru uvedena v poli Hobj v rámci volání MQSUB a nejedná se o spravovanou obslužnou rutinu, bude priorita zprávy převzata z atributu DefProirity této fronty. Je-li takto označená fronta fronta klastru nebo existuje více než jedna definice v cestě rozpoznání názvu fronty, je priorita určena při vložení zprávy publikování do fronty, jak je popsáno pro položku Priorita v deskriptoru MQMD. Pokud volání MQSUB používá spravovanou obslužnou rutinu, je priorita zprávy převzata z atributu DefPriority fronty modelu přidružené k odběru tématu přihlášeným k odběru.
- MQPRI_PRIORITY_AS_PUBLISHED-Priorita pro zprávu je priorita původní publikace. Toto je počáteční hodnota tohoto pole.

SubCorrelId

Upozornění: Identifikátor korelace může být předáván pouze mezi správci front v klastru publikování/odběru, ne v hierarchii.

Všechny publikace odeslané tak, aby odpovídaly tomuto odběru, budou obsahovat tento korelační identifikátor v deskriptoru zpráv. Pokud více odběrů používá stejnou frontu k získání svých publikací, použití funkce MQGET podle ID korelace umožňuje získat pouze publikování pro specifický odběr. Tento korelační identifikátor může vygenerovat buď správce front, nebo uživatel.

Není-li určena volba MQSO_SET_CORREL_ID, je identifikátor korelace generován správcem front a toto pole je výstupní pole, které obsahuje identifikátor korelace, který bude nastaven v každé zprávě publikované pro tento odběr.

Je-li určena volba MQSO_SET_CORREL_ID, je identifikátor korelace generován uživatelem a toto pole je vstupní pole, které obsahuje identifikátor korelace, který má být nastaven v každé publikaci pro tento odběr. V tomto případě, pokud pole obsahuje MQCI_NONE, bude korelační identifikátor, který bude nastaven v každé zprávě publikované pro tento odběr, představovat korelační identifikátor vytvořený původním vložením této zprávy.

Je-li zadána volba MQSO_GROUP_SUB a zadaný identifikátor korelace je shodný s existujícím seskupeným odběrem s použitím stejné fronty a překrývajícím se řetězcem tématu, je k dispozici pouze nejvýznamnější odběr ve skupině s kopií této publikace.

SubUserData

Jedná se o uživatelská data odběru. Data poskytnutá na odběru v tomto poli budou zahrnuta jako vlastnost datové zprávy MQSubUserpro každou publikaci odeslanou do tohoto odběru.

Vlastnosti publikování

Tabulka 44 na stránce 296 obsahuje seznam vlastností publikování, které jsou k dispozici spolu se zprávou o publikování.

K těmto vlastnostem můžete přistupovat přímo ze složky **MQRFH2** , nebo je načíst pomocí produktu MQINQMP. MQINQMP přijímá buď název vlastnosti, nebo **MQRFH2** název jako název vlastnosti, na kterou se má dotaz dotázat.

Tabulka 44. Vlastnosti publikování

Název vlastnosti	Název MQRFH2	Typ	Popis
MQTopicString	mqps.Top	MQTYPE_STRING	Řetězec tématu
MQSubUserData	mqps.Sud	MQTYPE_STRING	Uživatelská data odběratele
MQIsRetained	mqps.Ret	MQTYPE_BOOLEAN	Zachované publikování
MQPubOptions	mqps.Pub	MQTYPE_INT32	Volby publikování
MQPubLevel	mqps.Pbl	MQTYPE_INT32	Úroveň zveřejnění
MQPubTime	mqpse.Pts	MQTYPE_STRING	Čas publikování
MQPubSeqNum	mqpse.Seq	MQTYPE_INT32	Pořadové číslo publikace
MQPubStrIntData	mqpse.Sid	MQTYPE_STRING	String/Integer data přidaná vydavatelem
MQPubFormat	mqpse.Pfmt	MQTYPE_INT32	Formát zprávy: MQRFH1 MQRFH2 PCF

Uspořádání zpráv

U konkrétního tématu jsou zprávy publikovány správcem front ve stejném pořadí, v jakém jsou přijímány z publikování aplikací (je-li změna pořadí založena na prioritě zpráv).

Řazení zpráv obvykle znamená, že každý odběratel přijímá zprávy od konkrétního správce front v konkrétním tématu od určitého vydavatele v pořadí, ve kterém jsou publikovány vydavatelem.

Nicméně stejně jako u všech zpráv produktu WebSphere MQ je možné zprávy občas doručit mimo pořadí. K tomu může dojít v následujících situacích:

- Je-li odkaz v síti vypnutý a následné zprávy jsou přesměrovány podél jiného odkazu
- Pokud je fronta dočasně zaplněna nebo zablokována tak, aby byla zpráva vložena do fronty nedoručených zpráv, a proto zpožděna, zatímco následné zprávy procházejí přímo.
- Pokud administrátor odstraní správce front, když jsou vydavatelé a odběratelé stále v provozu, způsobí, že zprávy ve frontě budou umístěny do fronty nedoručených zpráv a odběry mají být přerušeny.

Pokud tyto okolnosti nemohou nastat, jsou publikace vždy doručovány v uvedeném pořadí.

Poznámka: Seskupené nebo segmentované zprávy nelze použít s publikováním/odběrem.

Zachycení publikací

Můžete zachytit publikování, upravit jej a pak ji znovu publikovat, než dosáhne dalšího odběratele.

Možná budete chtít před tím, než dojde k odběrateli, zachytit publikování, aby bylo možné provést jednu z následujících akcí:

- Připojit další informace ke zprávě
- Blokovat zprávu
- Transformovat zprávu

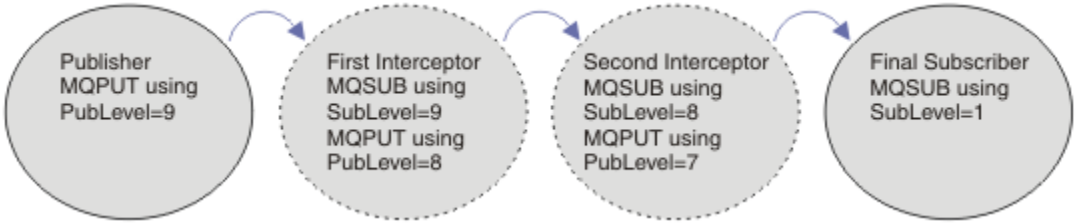
Na každé zprávě můžete provést stejnou operaci nebo můžete operaci změnit, a to v závislosti na odběru, zprávě nebo záhlaví zprávy.

Související odkazy

MQ_PUBLISH_EXIT-Uživatelská procedura publikování

Úrovně odběru

Nastavte úroveň odběru odběru, aby bylo možné publikování zachytit, než dosáhne svých konečných odběratelů. Zachytávající odběratel odebírá odběr na vyšší úrovni odběru a publikuje se na nižší úrovni publikování. Sestavte řetězec zachycující odběratele, aby před doručením koncovým odběratelům zpracovali zpracování zpráv na publikování.



Obrázek 59. Posloupnost zachytávajících odběratelů

Chcete-li zachytit publikování, použijte atribut **MQSD** `SubLevel` . Po zachycení zprávy lze tuto zprávu transformovat a poté znovu publikovat na nižší úrovni publikování změnou atributu **MQPMO** `PubLevel` . Zpráva pak přejde ke koncovým odběratelům, nebo je opět zadržena zprostředkujícím odběratelem na nižší úrovni odběru.

Odběratel zachycení obvykle transformuje zprávu před opětovným publikováním. Sled zpráv tvoří posloupnost zachycujících odběratelů. Eventuálně nelze znovu publikovat zachycenou publikaci: Odběratelé na nižších úrovních odběru by zprávu neobdrželi.

Ujistěte se, že zachytávač přijímá publikování před všemi ostatními odběrateli. Nastavte úroveň odběru zachytávače vyšší než ostatní odběratelé. Odběratelé standardně mají `SubLevel` produktu 1. Nejvyšší hodnota je 9. Publikování musí začínat znakem `PubLevel` alespoň stejně, jako nejvyšší `SubLevel`. Publikovat nejprve s výchozí hodnotou `PubLevel` produktu 9.

- Máte-li jedno zachycení odběratele na téma, nastavte parametr `SubLevel` na hodnotu 9.
- Pro více zakročujících aplikací na téma nastavte nižší úroveň `SubLevel` pro každého následného zachytávače zachycení.
- Můžete implementovat maximum 8 zachytávajících aplikací, s úrovněmi odběrů od 9 do 2 včetně. Konečný příjemce zprávy má `SubLevel` produktu 1.

Zachytávač s nejvyšší úrovní odběru, která se rovná nebo je nižší než `PubLevel` publikování, obdrží publikování jako první. Konfigurovat pouze jeden zachytávač odběratele pro téma na konkrétní úrovni odběru. Existence více odběratelů na určité úrovni odběru vede k více kopiím publikace zasílané do konečného souboru odebírajících aplikací.

Odběratel s hodnotou SubLevel produktu 0 se používá jako catchall. Pokud žádný koncový odběratel nedostane zprávu, obdrží tuto příručku. Odběratel s SubLevel produktu 0 může být použit k monitorování publikací, které neobdrželi jiní odběratelé.

Programování zachytávače zachycení

Použijte volby odběru popsané v tématu Tabulka 45 na stránce 298.

Tabulka 45. Volby odběru pro zachytávající odběratele	
Volba odběru	**Notes**
MQSO_SET_CORREL_ID a SubCorrelId jsou nastaveny na MQCI_NONE	Ponechte CorrelId zachycené publikace stejné jako původní publikování. **Poznámka:** Nelze předat identifikátor korelace publikování v hierarchii. Pole je používáno správcem front.
PubPriority je nastavena na MQPRI_PRIORITY_AS_PUBLISHED	Ponechejte prioritu zachycené publikace stejně jako v původní publikaci.

Volby v produktu Tabulka 45 na stránce 298 musí být použity všemi zachytávajícími odběrateli. Výsledkem je, že identifikátor korelace a priorita zprávy nejsou upraveny z nastavení původního vydavatele.

Když zachytávající odběratel zpracoval publikaci, znovu publikuje zprávu do stejného tématu na úrovni PubLevel o jednu nižší, než je SubLevel vlastního odběru. Pokud zachycovací odběratel nastavil SubLevel produktu 9, publikuje zprávu znovu s parametrem PubLevel produktu 8.

Chcete-li zprávu znovu publikovat správně, je třeba použít několik informací z původní příručky. Znovu použijte stejné **MQMD** jako v původní zprávě a nastavte MQPMO_PASS_ALL_CONTEXT , abyste se ujistili, že všechny informace v **MQMD** jsou předány dalšímu odběrateli. Zkopírujte hodnoty z vlastností zprávy, které jsou zobrazeny v Tabulka 46 na stránce 298 , do odpovídajících polí znovu publikované zprávy. Odběratel zachycení může tyto hodnoty změnit. Použijte operátor OR k přidání dalších hodnot do pole **MQPMO**.Volby , chcete-li kombinovat volby vkládání zpráv.

Frontu publikování musíte explicitně otevřít raději, než použít spravovanou frontu publikování. Pro spravovanou frontu nelze nastavit MQSO_SET_CORREL_ID . Také nelze nastavit MQOO_SAVE_ALL_CONTEXT ve spravované frontě. Viz fragmenty kódu uvedené v části "Příklady" na stránce 299.

Tabulka 46. Hodnoty MQPUT pro znovu publikované zprávy	
Znovu publikovat zprávu pomocí MQPUT	**Informace ve zprávě o publikování**
MQOD.ObjectString	vlastnost zprávy MQTopicString
MQPMO.Options	vlastnost zprávy MQPubOptions

Konečný odběratel má volbu nastavení voleb odběru rozdílně. Například, může nastavit prioritu publikování explicitně spíše než na MQPRI_PRIORITY_AS_PUBLISHED. Nastavení konečného odběratele ovlivní pouze publikování z konečného zachytávače odběratele v řetězci.

Zachovaná publikování

Zachované publikování musí být po zachycení uchováno zkopírováním původních voleb vložení zpráv do znovu publikované zprávy.

Volba MQPMO_RETAIN je nastavena vydavatelem. Každý zachytávací odběratel musí přenést MQPubOptions do voleb vložení zprávy znovu publikovaná zpráva, jak ukazuje Tabulka 46 na stránce

298. Kopírování voleb vkládání zpráv zachovává volby nastavené původním vydavatelem, včetně informací o tom, zda má být publikace zachována.

Jakmile publikace dokončí svůj průchod řetězu zachycení odběratelů a doručí se konečným odběratelům, je nakonec uchován. Noví odběratelé, na `SubLevel 1`, požadující zachované publikování, jej přijmou bez dalšího zaceňování. Odběratelé na `SubLevel` větší než 1 neodešlou zachované publikování. V důsledku toho zůstává zachované publikování neupravováno řetězcem zachycení odběratele podruhé zaokrouhlením.

Příklady

Příklady jsou fragmenty kódu, které lze zkombinovat k sestavení syndikátu odběratele. Kodex je napsán jako stručný, spíše než v kvalitě výroby.

Direktivy preprocesoru v produktu Obrázek 60 na stránce 299 definují dvě vlastnosti, které mají být extrahovány ze zpráv publikování, které jsou vyžadovány voláním MQI produktu `MQINQMP` .

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <cmqc.h>
#define      MQPUBOPTIONS        (MQPTR)(char*) "MQPubOptions",\
                                 0,\
                                 12,\
                                 MQVS_NULL_TERMINATED,\
                                 MQCCSI_APPL
#define      MQTOPICSTRING       (MQPTR)(char*) "MQTopicString",\
                                 0,\
                                 13,\
                                 MQVS_NULL_TERMINATED,\
                                 MQCCSI_APPL
```

Obrázek 60. Direktivy preprocesoru

Příkaz Obrázek 61 na stránce 300 vypíše deklarace použité v fragmentech kódu. S výjimkou zvýrazněných výrazů jsou deklarace standardní pro aplikaci WebSphere MQ .

Zvýrazněné volby Put a Get jsou inicializovány tak, aby prošly celý kontext. Zvýrazněné moduly `MQTOPICSTRING` a `MQPUBOPTIONS` jsou inicializátory MQCHARV pro názvy vlastností, které jsou definovány v direktivách preprocesoru. Názvy jsou předány produktu `MQINQMP`.

```
int main(int argc, char **argv) {
    MQLONG   Reason            = MQRC_NONE;
    MQLONG   CompCode          = MQCC_OK;
    MQHCONN  Hcon              = MQHC_UNUSABLE_HCONN;
    MQCHAR   QMName[49]        = " ";
    MQCMHO   CrtMsgHOpts       = {MQCMHO_DEFAULT};
    MQHMSG   Hmsg              = MQHM_NONE;
    MQMD     md                = {MQMD_DEFAULT};
    MQHOBJ   gHobj             = MQHO_NONE;
    MQOD     getOD             = {MQOD_DEFAULT};
    MQGMO    gmo               = {MQGMO_DEFAULT};
    MQLONG   GO_Options        = MQOO_INPUT_AS_Q_DEF
                               | MQOO_FAIL_IF_QUIESCING
                               | MQOO_SAVE_ALL_CONTEXT;
    MQLONG   GC_Options        = MQCO_DELETE_PURGE;
    MQHOBJ   Hsub              = MQHO_NONE;
    MQSD     sd                = {MQSD_DEFAULT};
    MQLONG   SC_Options        = MQCO_NONE;
    MQHOBJ   pHobj             = MQHO_NONE;
    MQOD     putOD             = {MQOD_DEFAULT};
    MQLONG   PO_Options        = MQOO_OUTPUT
                               | MQOO_FAIL_IF_QUIESCING
                               | MQOO_PASS_ALL_CONTEXT;
    MQLONG   PC_Options        = MQCO_NONE;
    MQPMO    pmo               = {MQPMO_DEFAULT};
    MQIMPO   InqPropOpts       = {MQIMPO_DEFAULT};
    MQPD     PropDesc          = {MQPD_DEFAULT};
    MQLONG   Type              = MQTYPE_AS_SET;
    MQCHARV  TopStrProp        = {MQTOPICSTRING};
    MQCHARV  PubOptProp        = {MQPUBOPTIONS};
    MQLONG   DataLength        = 0;
    MQBYTE   buffer[256]       = "";
    MQLONG   buflen            = sizeof(buffer) - 1;
    MQLONG   messlen           = 0;
    char     TopStrBuf[256]    = "Initial value";
    int      i                 = 0;
```

Obrázek 61. Deklarace

Inicializace, které nejsou snadno provedeny v deklaracích, jsou zobrazeny v Obrázek 62 na stránce 301. Zvýrazněné hodnoty vyžadují vysvětlení.

SYSTEM.NDURABLE.MODEL.QUEUE
V tomto příkladu místo použití produktu MQSUB pro otevření spravovaného netrvalého odběru se modelová fronta SYSTEM.NDURABLE.MODEL.QUEUEpoužívá k vytvoření dočasné dynamické fronty. Jeho popisovač se předá do MQSUB. Otevřením fronty přímo budete schopni uložit celý kontext zprávy a nastavit volbu odběru MQSO_SET_CORREL_ID.

MQGMO_CURRENT_VERSION
Je důležité používat aktuální verzi většiny struktur produktu WebSphere MQ . Pole jako gmo.MsgHandle jsou k dispozici pouze v nejnovější verzi řídicích struktur.

MQGMO_PROPERTIES_IN_HANDLE
Řetězec tématu a volby vložení zpráv nastavené v původní publikaci mají být načteny zachycujícím odběratelem pomocí vlastností zprávy. Alternativou by bylo čtení struktury **MQRFH2** přímo ve zprávě.

MQSO_SET_CORREL_ID
Použijte MQSO_SET_CORREL_ID v kombinaci s,

```
memcpy(sd.SubCorrelId, MQCI_NONE, sizeof(sd.SubCorrelId));
```

Efekt těchto voleb je předání identifikátoru korelace. Identifikátor korelace nastavený původním vydavatelem je umístěn v poli identifikátoru korelace v publikování, které přijímá zachytávačovatel zachycení. Každý zachytávající odběratel přechází na stejný identifikátor korelace. Konečný odběratel pak má volbu přijetí stejného identifikátoru korelace.

Poznámka: Je-li publikace předávána prostřednictvím hierarchie publikování/odběru, nebude identifikátor korelace nikdy zachován.

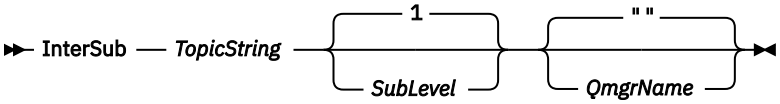
MQPRI_PRIORITY_AS_PUBLISHED
Publikace se umístí do fronty publikování se stejnou prioritou zprávy, jako byla publikována.

```
    strncpy(getOD.ObjectName,    "SYSTEM.NDURABLE.MODEL.QUEUE",
            sizeof(getOD.ObjectName));
    gmo.Version                = MQGMO_VERSION_4;
    gmo.Options                = MQGMO_WAIT
                               | MQGMO_PROPERTIES_IN_HANDLE
                               | MQGMO_CONVERT;
    gmo.WaitInterval           = 30000;
    sd.Options                 = MQSO_CREATE
                               | MQSO_FAIL_IF_QUIESCING
                               | MQSO_SET_CORREL_ID;
    sd.PubPriority             = MQPRI_PRIORITY_AS_PUBLISHED;
    sd.Version                 = MQSD_VERSION_1;
    memcpy(sd.SubCorrelId, MQCI_NONE, sizeof(sd.SubCorrelId));
    putOD.ObjectType             = MQOT_TOPIC;
    putOD.ObjectString.VSPtr     = &TopStrBuf;
    putOD.ObjectString.VSBufSize = sizeof(TopStrBuf);
    putOD.ObjectString.VSLength  = MQVS_NULL_TERMINATED;
    putOD.ObjectString.VSCCSID   = MQCCSI_APPL;
    putOD.Version                = MQOD_VERSION_4;
    pmo.Version                = MQPMO_VERSION_3;
```

Obrázek 62. Inicializace

Obrázek 63 na stránce 302 ukazuje fragment kódu pro čtení parametrů příkazového řádku, dokončení inicializace a vytvoření odběru odběru.

Spusťte program s příkazem,



Aby zpracování chyb bylo možné bez omezení, je kód příčiny každého volání MQI uložen v odlišném prvku pole. Po každém volání kódu dokončení je testován kód dokončení a pokud je hodnota MQCC_FAIL, ovládací prvek ukončí blok kódu do `{ } while(0)`.

Dva zajímavé řádky kódu jsou,

pmo.PubLevel = sd.SubLevel - 1;
: Nastaví úroveň publikování pro znovu publikovanou zprávu na nižší než úroveň odběru zachycujícího odběratele.

gmo.MsgHandle = Hmsg;
: Poskytuje popisovač zprávy pro MQGET , aby vrátil vlastnosti zprávy.

```
do {
   printf("Intercepting subscriber start\n");
   if (argc < 2) {
       printf("Required parameter missing - topic string\n");
       exit(99);
   } else {
       sd.ObjectString.VSPtr    = argv[1];
       sd.ObjectString.VSLength = MQVS_NULL_TERMINATED;
       printf("TopicString = %s\n", sd.ObjectString.VSPtr);
   }
   if (argc > 2) {
       sd.SubLevel  = atoi(argv[2]);
       pmo.PubLevel = sd.SubLevel - 1;
       printf("SubLevel is %d, PubLevel is %d\n", sd.SubLevel, pmo.PubLevel);
   }
   if (argc > 3)
       strncpy(QMName, argv[3], sizeof(QMName));
   MQCONN(QMName, &Hcon, &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
       break;
   MQOPEN(Hcon, &getOD, GO_Options, &gHobj, &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
       break;
   MQSUB(Hcon, &sd, &gHobj, &Hsub, &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
       break;
   MQCRTMH(Hcon, &CrtMsgHOpts, &Hmsg, &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
       break;
   gmo.MsgHandle = Hmsg;
```

Obrázek 63. Příprava na zachycení publikování

Hlavní fragment kódu, Obrázek 64 na stránce 303, získává zprávy z fronty publikování. Dotazuje se na vlastnosti zprávy a znovu publikuje zprávy pomocí řetězce tématu a původních vlastností produktu **MQPMO**.option .

V tomto příkladu není na publikování provedena žádná transformace. Řetězec tématu publikované publikace se vždy shoduje s řetězcem tématu, k jehož odběru přihlášeno odběratele. Je-li zachytávající odběratel zodpovědný za zachycení více odběrů odeslaných do stejné publikační fronty, může být nutné dotaz na řetězec tématu odlišit od publikací, které odpovídají různým odběrům.

Volání na MQINQMP jsou zvýrazněna. Vlastnosti řetězce tématu a volby vložení vlastností zprávy jsou zapsány přímo do řídicích struktur výstupu. Jediný důvod změny pole s délkou MQCHARV souboru putOD.ObjectString z explicitní délky na řetězec s ukončenou hodnotou null má za účelem výstupu řetězce použít příkaz printf .

```
while (CompCode != MQCC_FAILED) {
   memcpy(md.MsgId, MQMI_NONE, sizeof(md.MsgId));
   memcpy(md.CorrelId, MQCI_NONE, sizeof(md.CorrelId));
   md.Encoding       = MQENC_NATIVE;
   md.CodedCharSetId = MQCCSI_Q_MGR;
   printf("MQGET : %d seconds wait time\n", gmo.WaitInterval/1000);
   MQGET(Hcon, gHobj, &md, &gmo, buflen, buffer, &messlen,
       &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
       break;
   buffer[messlen] = '\0';
   MQINQMP(Hcon, Hmsg,    &InqPropOpts, &TopStrProp, &PropDesc, &Type,
       putOD.ObjectString.VSBufSize, putOD.ObjectString.VSPtr,
       &(putOD.ObjectString.VSLength), &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
       break;
   memset((void *)((MQLONG)(putOD.ObjectString.VSPtr)
       + putOD.ObjectString.VSLength),'\0',1);
   putOD.ObjectString.VSLength = MQVS_NULL_TERMINATED;
   MQINQMP(Hcon, Hmsg, &InqPropOpts, &PubOptProp, &PropDesc, &Type,
       sizeof(pmo.Options), &(pmo.Options), &DataLength,
       &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
       break;
   MQOPEN(Hcon, &putOD, PO_Options, &pHobj, &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
       break;
   printf("Republish message <%s> on topic <%s> with options %d\n",
       buffer, putOD.ObjectString.VSPtr, pmo.Options);
   MQPUT(Hcon, pHobj, &md, &pmo, messlen, buffer, &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
       break;
   MQCLOSE(Hcon, &pHobj, PC_Options, &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
   break;
}
```

Obrázek 64. Publikování a opětovné publikování produktu Intercept

Poslední fragment kódu je zobrazen v Obrázek 65 na stránce 303.

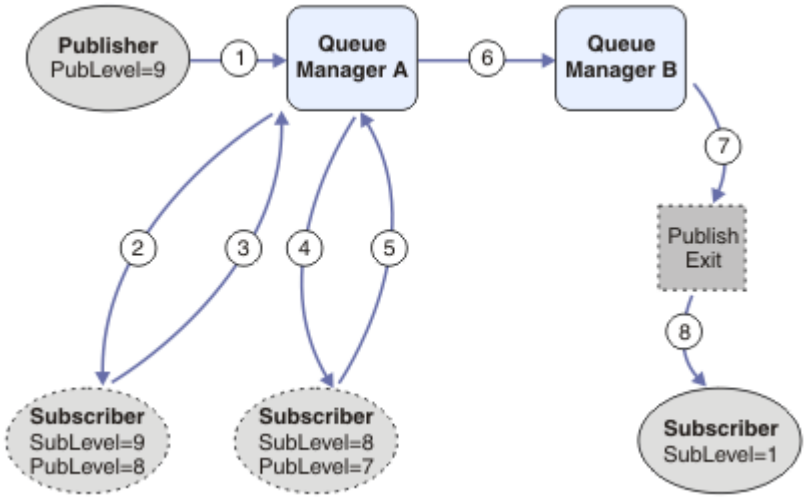
```
    } while (0);
    if (CompCode == MQCC_FAILED && Reason != MQRC_NO_MSG_AVAILABLE)
        printf("MQI Call failed with reason code %d\n", Reason);
    if (Hsub!= MQHO_NONE)
        MQCLOSE(Hcon, &Hsub, SC_Options, &CompCode, &Reason);
    if (Hcon!= MQHC_UNUSABLE_HCONN)
        MQDISC(&Hcon, &CompCode, &Reason);
}
```

Obrázek 65. Dokončení

Zachycení publikací a distribuovaného publikování/odběru

Při implementaci zachycení odběratelů nebo publikování v rámci distribuované topologie publikování/odběru postupujte podle jednoduchého vzoru. Implementace zachytává odběratele ve stejných správcích front jako vydavatelé a Publish opouští ve stejných správcích front jako koneční odběratelé.

Produkt Obrázek 66 na stránce 304 zobrazuje dva správce front připojené v klastru odběru publikování. Vydavatel vytvoří publikování na téma klastru na úrovni publikování 9. Očíslované šipky ukazují posloupnost kroků, které byly v publikování provedeny, když se přenášejí na odběratele v tématu klastru. Publikování je zachyceno odběratelem s hodnotou `Dílčí úroveň` 9 a znovu publikovány s úrovní `Publevel` 8. Odběratel je znovu zachycen na `Sublevel` 8. Odběratel znovu publikuje na `Publevel` 7. Odběratel proxy poskytnutý správcem front postoupí publikování do správce front B, kde byla kromě konečného odběratele implementována uživatelská procedura publikování. Publikování je zpracováno uživatelskou procedurou publikování před tím, než je nakonec přijata koncovým odběratelem na `dílčí úrovni` 1. Zachycovací odběratelé a uživatelská procedura publikování se zobrazí se zalomenými obrysy.



Obrázek 66. Výslech a uživatelská procedura publikování v klastru

Cílem jednoduchého vzoru je pro každého odběratele, který přijímá publikování, aby obdržel stejnou publikaci. Publikování probíhá prostřednictvím stejné posloupnosti transformací bez ohledu na to, kde je odběratel připojen. Pravděpodobně se chcete vyhnout posloupnosti transformací, a to v závislosti na tom, kde jsou vydavatelé nebo koneční odběratelé připojeni. Přípravou výjimkou by bylo přizpůsobení příručky konečnému uživateli každému jednotlivému odběrateli. Pomocí uživatelské procedury publikování lze přizpůsobit publikování na základě fronty, do níž je publikace konečně doručena.

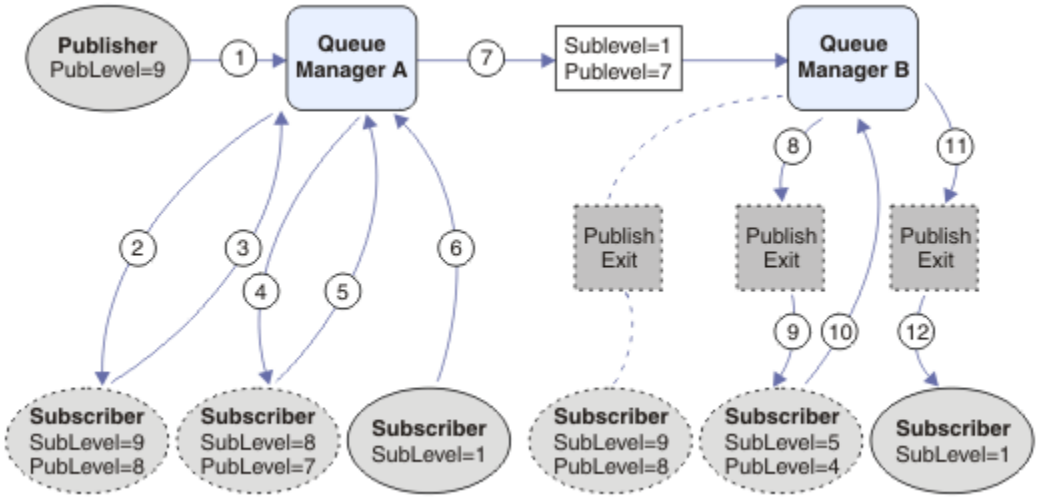
Musíte pečlivě zvážit, kam implementovat zachycení odběratele a ukončení publikování v topologii distribuovaného publikování/odběru. Přímočarý návrhový vzor implementuje zachycení odběratelů do stejného správce front jako vydavatelé a Publish exits ke stejným správcům front jako koneční odběratelé.

Anti-pattern

Obrázek 67 na stránce 305 ukazuje, jak se mohou věci navrátit, pokud nepostupujete podle jednoduchého vzoru. Chcete-li zkomplikovat implementaci, je do správce front A přidán konečný odběratel a do správce front B jsou přidáni dva další zachytávaní odběratelé.

Publikování bude předáno správci front B na úrovni `PubLevel` 7, kde je zachycen odběratelem na `SubLevel` 5 před spotřebním konečným odběratelem na `SubLevel` 1. Uživatelská procedura publikování zachycuje publikování před tím, než je předána do zachytávačů spotřebitele i u konečného spotřebitele ve správci front B. Publikování dosáhne konečného odběratele ve správci front A, aniž by bylo zpracováno uživatelskou procedurou publikování.

V topologii publikování/odběru se odběratelé proxy přihlásí k odběru na `SubLevel` 1a předávají na `PubLevel` nastaveni posledním zachycujícím odběratelem. V produktu Obrázek 67 na stránce 305je výsledkem skutečnost, že publikování není zachyceno odběratelem s použitím produktu `SubLevel` 9 ve správci front B.



Obrázek 67. Komplexní implementace zachycujících odběratelů

Volby publikování

K dispozici je několik možností, které řídí způsob, jakým jsou zprávy publikovány.

Srážková odpověď-na informace od odběratelů

Pokud nechcete, aby odběratelé mohli odpovědět na publikování, která přijímají, je možné zatajit informace v polích ReplyToQ a ReplyToQmgr deskriptoru MQMD pomocí volby put-message příkazu MQPMO_SUPPRESS_REPLYTO. Je-li tato volba použita, odebere správce front tyto informace z MQMD, jakmile obdrží publikování, než ji předá všem odběratelům.

Tuto volbu nelze použít v kombinaci s volbou sestavy, která potřebuje odpověď ReplyToQ, pokud se tento pokus o volání nezdaří s chybou MQRC_MISSING_REPLY_TO_Q.

Úroveň zveřejnění

Použití úrovní publikování je způsob, jak řídit, kteří odběratelé obdrží publikování. Úroveň publikování určuje úroveň odběru, na kterou se má publikace zaměřit. Publikaci obdrží pouze odběry s nejvyšší úrovní odběru, která je nižší než úroveň publikování nebo její úrovně zveřejnění. Tato hodnota musí být v rozsahu nula až devět; nula je nejnižší úroveň publikování. Počáteční hodnota tohoto pole je 9. Jednou z použití úrovní publikování a odběru je zachycení publikování.

Kontrola, zda není publikování doručeno žádnému odběratelům

Chcete-li zkontrolovat, zda publikování nebylo doručeno žádnému odběrateli, použijte volbu vložení zprávy MQPMO_WARN_IF_NO_SUBS_MATCHED s voláním MQPUT. Pokud operace vložení vrátí kód dokončení MQCC_WARNING a kód příčiny MQRC_NO_SUBS_MATCHED, publikování nebylo doručeno do žádných odběrů. Je-li v operaci vložení zadána volba MQPMO_RETAIN, bude zpráva uchována a doručena do všech následně definovaných odpovídajících odběrů. V distribuovaném systému publikování/odběru je návratový kód MQRC_NO_SUBS_MATCHED vrácen pouze v případě, že nejsou registrovány žádné odběry proxy pro dané téma ve správci front.

Volby odběru

K dispozici je několik možností, které řídí způsob zpracování odběrů zpráv.

Trvalost zpráv

Správci front udržují perzistenci publikací, které předávají odběratelům, jak je nastaveno vydavatelem. Vydavatel nastaví perzistenci na jednu z následujících možností:

0
Přechodné

1
Trvalý

2
Perzistence jako fronta/definice tématu

U publikování/odběru vydavatel vyřeší objekt tématu a produkt **topicString** na vyřešený objekt tématu. Pokud vydavatel určuje definici trvání jako frontu/téma, bude pro publikování nastavena výchozí perzistence z vyřešeného objektu tématu.

Zachovaná publikování

Pro řízení při přijetí zachovaných publikování mohou odběratelé používat dvě volby odběru:

Publikovat na žádost pouze, MQSO_PUBLICATIONS_ON_REQUEST

Chcete-li, aby odběratel měl kontrolu nad tím, kdy přijímá publikace, můžete použít volbu odběru MQSO_PUBLICATIONS_ON_REQUEST. Odběratel může poté řídit, kdy obdrží publikování, pomocí volání MQSUBRQ (zadání manipulátoru Hsub, který byl vrácen z původního volání MQSUB) k odeslání požadavku na jeho zachované publikování. Odběratelé používající volbu odběru MQSO_PUBLICATIONS_ON_REQUEST nepřijímají žádné nezachované publikace.

Uvedete-li MQSO_PUBLICATIONS_ON_REQUEST, musíte použít MQSUBRQ k načtení libovolné publikace. Pokud nepoužíváte funkci MQSO_PUBLICATIONS_ON_REQUEST, získáte zprávy při jejich publikování.

Pokud odběratel používá volání MQSUBRQ a používá zástupné znaky v tématu odběru, může odběr odpovídat více tématům nebo uzlům ve stromu témat, přičemž všechny zachované zprávy (pokud nějaké existují) budou odeslány odběrateli.

Tato volba může být užitečná zejména při použití s trvalým odběrem, protože správce front bude i nadále odesílat publikace odběrateli, pokud je trvale odebírán, a to i v případě, že aplikace odběratele není spuštěna. To by mohlo vést k hromadným zprávám ve frontě odběratele. Tomuto sestavení lze se vyhnout, pokud se odběratel registruje pomocí volby MQSO_PUBLICATIONS_ON_REQUEST. Případně můžete použít netrvalé odběry, pokud je to vhodné pro vaši aplikaci, abyste se vyvarovali vytváření nežádoucích zpráv.

Je-li odběr trvalý a vydavatel používá zachované publikace, může aplikace odběratele použít volání MQSUBRQ k aktualizaci informací o stavu po restartu. Odběratel musí poté pravidelně aktualizovat svůj stav pomocí volání MQSUBRQ.

Pomocí této volby nebudou odeslány žádné publikace jako výsledek volání MQSUB. Trvalý odběr, který byl obnoven po odpojení, bude používat volbu MQSO_PUBLICATIONS_ON_REQUEST, pokud byl pro použití této volby konfigurován původní odběr.

Pouze nové publikování, pouze MQSO_NEW_PUBLICATIONS_ONLY

Pokud v tématu existuje zachované publikování, obdrží všechny odběratele, kteří provádějí odběr po publikování, kopii této publikace. Pokud odběratel nechce přijímat žádná publikování, která byla provedena dříve, než je předplatný odběr, může odběratel použít volbu odběru MQSO_NEW_PUBLISATIONS_ONLY.

Seskupení odběrů

Uvažte seskupení odběrů, pokud jste nastavili frontu pro příjem publikování a máte-li počet překrývajících se odběrů, které dodávají do stejné fronty. Tato situace je podobná jako příklad v tématu Překrývání odběrů.

Když se přihlásíte k odběru tématu, můžete se vyhnout příjmu duplicitních publikování nastavením volby MQSO_GROUP_SUB . Výsledkem je, že pokud se více než jeden odběr ve skupině shoduje s tématem

publikování, odpovídá za umístění publikování do fronty pouze jeden odběr. Další odběry, které se shodují s tématem publikování, jsou ignorovány.

Odběr, který je odpovědný za umístění publikování do fronty, je vybrán na základě toho, že má nejdelší odpovídající řetězec tématu, než se narazí na libovolné zástupné znaky. Může být považováno za nejbližší odpovídající odběr. Jeho vlastnosti jsou šířeny do publikování, včetně toho, zda má vlastnost `MQSO_NOT_OWN_PBS` . Pokud ano, nebude do fronty doručena žádná publikace, i když jiná odpovídající předplatná nemusí mít vlastnost `MQSO_NOT_OWN_PUBS` .

Nemůžete umístit všechny své odběry do jedné skupiny, abyste eliminovali duplicitní publikace. Seskupené odběry musí splňovat tyto podmínky:

1. Žádná z odběrů není spravována.
2. Skupina odběrů doručují publikace do stejné fronty.
3. Každý odběr musí být na stejné úrovni odběru.
4. Zpráva o publikování pro každý odběr ve skupině má stejný korelační identifikátor.

   Chcete-li zajistit, aby každý odběr měl ve zprávě publikování se stejným identifikátorem korelace, nastavte parametr `MQSO_SET_CORREL_ID` pro vytvoření vlastního identifikátoru korelace v rámci publikování a nastavte stejnou hodnotu v poli **SubCorrelId** v každém odběru. Nenastavujte **SubCorrelId** na hodnotu `MQCI_NONE`.

Další informace viz ../com.ibm.mq.ref.dev.doc/q100080_.dita#q100080_/mqso_group_sub .

Inquaning about a nastavení atributů objektu

Atributy jsou vlastnosti, které definují charakteristiky objektu produktu WebSphere MQ .

Dotýkají se způsobu, jakým správce front zpracovává objekt. Atributy každého typu objektu produktu WebSphere MQ jsou podrobně popsány v tématu Atributy objektů.

Některé atributy jsou nastaveny při definování objektu a lze je měnit pouze pomocí příkazů produktu WebSphere MQ ; příklad takového atributu je výchozí prioritou pro zprávy zařazené do fronty. Ostatní atributy jsou ovlivněny operací správce front a mohou se v průběhu času měnit; příkladem je aktuální hloubka fronty.

Můžete se dotázat na aktuální hodnoty většiny atributů pomocí volání MQINQ. Modul MQI také poskytuje volání MQSET, pomocí kterého můžete změnit některé atributy fronty. Volání MQI nelze použít ke změně atributů žádného jiného typu objektu. Místo toho musíte použít:

Pro platformy WebSphere MQ pro platformy Windows, UNIX a Linux

Prostředek MQSC, který je popsán v příručce MQSC reference.

Poznámka: Názvy atributů objektů jsou zobrazeny v této dokumentaci v podobě, kterou používáte s voláními MQINQ a MQSET. Použijete-li příkazy produktu WebSphere MQ k definování, úpravě nebo zobrazení atributů, musíte identifikovat atributy pomocí klíčových slov zobrazených v popisech příkazů v odkazech témat.

Volání MQINQ i volání MQSET používají pole selektorů k identifikacity atributy, které chcete dotázat nebo nastavit. Pro každý atribut, se kterým můžete pracovat, je selektor. Název selektoru má předponu určenou charakterem atributu:

MQCA_	Tyto selektory odkazují na atributy, které obsahují znaková data (například název fronty).
MQIA_	Tyto selektory odkazují na atributy, které obsahují buď číselné hodnoty (jako například *CurrentQueueDepth*, počet zpráv ve frontě) nebo konstantní hodnotu (jako například *SyncPoint*, zda správce front podporuje synchronizační body).

Před použitím volání MQINQ nebo MQSET musí být vaše aplikace připojena ke správci front a vy musíte použít volání MQOPEN k otevření objektu pro nastavení nebo zjišťování informací o atributech. Tyto

operace jsou popsány v “Připojování k správci front a odpojování od něj” na stránce 198 a “Otevírání a zavírání objektů” na stránce 206.

Následující odkazy vám pomohou zjistit další informace o získávání dotazu a nastavení atributů objektu:

- “Inquaktování o atributech objektu” na stránce 308
- “Některé případy, kdy volání MQINQ selže” na stránce 309
- “Nastavení atributů fronty” na stránce 309

Související pojmy

“Přehled rozhraní fronty zpráv” na stránce 187
Zde jsou uvedeny informace o komponentách rozhraní MQI (Message Queue Interface).

“Připojování k správci front a odpojování od něj” na stránce 198
Chcete-li používat programovací služby produktu WebSphere MQ , musí mít program připojení ke správci front. V této části se dozvíte, jak se připojit k správci front a odpojit je od správce front.

“Otevírání a zavírání objektů” na stránce 206
Tyto informace poskytují náhled na otevření a zavření objektů produktu WebSphere MQ .

“Vložení zpráv do fronty” na stránce 216
V této části se dozvíte, jak vložit zprávy do fronty.

“Získávání zpráv z fronty” na stránce 231
Tyto informace použijte k získání informací o získávání zpráv z fronty.

“Potvrzení a zálohování jednotek práce” na stránce 310
Tyto informace popisují, jak potvrdit a zazálohovat všechny zotavitelné operace get a put, které se vyskytly v jednotce práce.

“Spuštění aplikací produktu IBM WebSphere MQ pomocí spouštěčů” na stránce 316
Informace o spouštěčích a o tom, jak spustit aplikace IBM WebSphere MQ pomocí spouštěčů.

“Práce s MQI a klastry” na stránce 332
Existují speciální volby na voláních a návratových kódech, které se vztahují ke klastrování.

Inquaktování o atributech objektu

Použijte volání MQINQ k dotazům na atributy libovolného typu IBM WebSphere MQ.

Jako vstup pro toto volání musíte dodat:

- Popisovač připojení.
- Popisovač objektu.
- Počet selektorů.
- Pole selektorů atributů, každý selektor má tvar MQCA_ * nebo MQIA_ *. Každý selektor představuje atribut s hodnotou, o které se chcete dotázat, a každý selektor musí být platný pro typ objektu, který popisovač objektu představuje. Selektory můžete určit v libovolném pořadí.
- Počet celočíselných atributů, o které se budete dotazovat. Uveďte nulu, pokud se nedotazujete na celočíselné atributy.
- Délka vyrovnávací paměti znakových atributů v *CharAttrLength*. Musí to být alespoň součet délek požadovaných k zadržení každého znakového řetězce atributu. Uveďte nulu, pokud se nedotazujete na znakové atributy.

Výstup z MQINQ je:

- Sada celočíselných hodnot atributů kopírovaných do pole. Počet hodnot je určen produktem *IntAttrCount*. Pokud je hodnota *IntAttrCount* nebo *SelectorCount* nula, tento parametr se nepoužije.
- Vyrovnávací paměť, ve které jsou vráceny znakové atributy. Délka vyrovnávací paměti je dána parametrem *CharAttrLength* . Pokud je hodnota *CharAttrLength* nebo *SelectorCount* nula, tento parametr se nepoužije.

- Kód dokončení. Pokud kód dokončení vydá varování, znamená to, že volání bylo dokončeno pouze částečně. V takovém případě zkontrolujte kód příčiny.
- Kód příčiny. K dispozici jsou tři situace s částečnými dokončením:
  - Selektor není použit pro daný typ fronty.
  - Pro celočíselné atributy není k dispozici dostatek místa.
  - Pro atributy znaků není k dispozici dostatek místa.

  Pokud nastane více než jedna z těchto situací, bude vrácena první z těchto situací.

Pokud otevřete frontu pro výstup nebo dotaz a přeloží se na nemístnou frontu klastru, můžete se dotázat pouze na název fronty, typ fronty a společné atributy. Hodnoty obecných atributů jsou hodnoty zvolené fronty, pokud byla použita hodnota MQOO_BIND_ON_OPEN. Hodnoty jsou hodnoty libovolné z možných front klastru, pokud bylo použito buď MQOO_BIND_NOT_FIXED nebo MQOO_BIND_ON_GROUP, nebo MQOO_BIND_AS_Q_DEF, a atribut fronty *DefBind* byl MQBND_BIND_NOT_FIXED. Další informace viz “MQOPEN a klastry” na stránce 333 a MQOPEN .

Poznámka: Hodnoty vrácené voláním jsou snímkem vybraných atributů. Tyto atributy se mohou změnit dříve, než se váš program bude chovat na vrácených hodnotách.

V adresáři MQINQje uveden popis volání MQINQ.

Některé případy, kdy volání MQINQ selže

Pokud otevřete alias k dotazům na jeho atributy, vrátíte se k atributům alias fronty (objekt WebSphere MQ používaný pro přístup k jiné frontě), nikoli k základní frontě.

Avšak definice základní fronty, na kterou se alias řeší, je také otevřena správcem front, a pokud jiný program změní použití základní fronty v intervalu mezi voláními MQOPEN a MQINQ, volání MQINQ selže a vrátí kód příčiny MQRC_OBJECT_CHANGED. Volání také selže, jestliže se změní atributy objektu alias fronty.

Podobně platí, že když otevřete vzdálenou frontu a dotáže se na její atributy, vrátíte se pouze na atributy lokální definice vzdálené fronty.

Uvedete-li jeden nebo více selektorů, které nejsou platné pro typ atributů fronty, na které jste se dotazovali, volání MQINQ se dokončí s varováním a nastaví výstup následujícím způsobem:

- Pro celočíselné atributy jsou nastaveny odpovídající prvky proměnné *IntAttrs* na hodnotu MQIAV_NOT_APPLICABLE.
- Pro znakové atributy jsou odpovídající části řetězce *CharAttrs* nastaveny na hvězdičky.

Uvedete-li jeden nebo více selektorů, které nejsou platné pro typ atributů objektu, na které jste se dotazovali, volání MQINQ selže a vrátí kód příčiny MQRC_SELECTOR_ERROR.

Nelze volat MQINQ k zobrazení modelové fronty; lze použít buď prostředek MQSC, nebo příkazy dostupné na vaší platformě.

Nastavení atributů fronty

V této části se dozvíte, jak nastavit atributy fronty pomocí volání MQSET.

Pomocí volání MQSET můžete nastavit pouze následující atributy fronty:

- *InhibitGet* (ale ne pro vzdálené fronty)
- *DistList* (ne v systému z/OS)
- *InhibitPut*
- *TriggerControl*
- *TriggerType*
- *TriggerDepth*
- *TriggerMsgPriority*

- *TriggerData*

Volání MQSET má stejné parametry jako volání MQINQ. Pro funkci MQSET jsou však všechny parametry s výjimkou kódu dokončení a kódu příčiny vstupní parametry. K dispozici nejsou žádné situace s částečnými dokončením.

Poznámka: Rozhraní MQI nelze použít k nastavení atributů objektů produktu WebSphere MQ jiných než lokálně definovaných front.

Další informace o volání MQSET naleznete v části MQSET.

Potvrzení a zálohování jednotek práce

Tyto informace popisují, jak potvrdit a zazálohovat všechny zotavitelné operace get a put, které se vyskytly v jednotce práce.

V tomto tématu jsou použity následující termíny:

- Potvrdit
- Vrátit zpět
- Koordinace synchronizačních bodů
- Synchronizační bod
- Pracovní jednotka
- jednofázové potvrzení
- Dvoufázové potvrzení

Pokud jste obeznámeni s těmito podmínkami zpracování transakcí, můžete přejít k tématu “Aspekty synchronizačních bodů v aplikacích produktu IBM WebSphere MQ” na stránce 311.

Potvrdit a odvrátit

Když program vloží zprávu do fronty v rámci pracovní jednotky, tato zpráva se zviditelní ostatním programům pouze v případě, že program potvrdí jednotku práce. Chcete-li potvrdit jednotku práce, musí být všechny aktualizace úspěšné, aby se zachovala integrita dat. Pokud program zjistí chybu a rozhodne se, že operace vložení není trvalá, může vrátit jednotku práce zpět. Když program provádí odvolání, produkt IBM WebSphere MQ obnoví frontu odebráním zpráv, které byly vloženy do fronty touto jednotkou práce. Způsob, jakým program provádí operace commit a back out, závisí na prostředí, ve kterém je program spuštěn.

Podobně platí, že když program dostane zprávu z fronty v rámci pracovní jednotky, zůstává tato zpráva ve frontě, dokud program nepotvrdí jednotku práce, ale zpráva není k dispozici pro načtení jinými programy. Zpráva je trvale odstraněna z fronty, když program potvrdí jednotku práce. Pokud program zálohuje pracovní jednotku, produkt IBM WebSphere MQ obnoví frontu tím, že zpřístupní zprávy, které mají být načteny jinými programy.

Koordinace synchronizačních bodů, synchronizační bod, jednotka práce

Koordinace synchronizačních bodů je proces, při kterém jsou jednotky práce buď potvrzeny, nebo zálohovány s integritou dat.

Rozhodnutí o provedení změn nebo odvolání změn se provádí v nejjednodušším případě na konci transakce. Může však být užitečnější, aby aplikace synchronizovala změny dat na jiných logických bodech v rámci transakce. Tyto logické body se nazývají *synchronizační body* (nebo *synchronizační body*) a období zpracování sady aktualizací mezi dvěma synchronizami se nazývá *jednotka práce*. Několik volání MQGET a volání MQPUT může být součástí jediné jednotky práce. Maximální počet zpráv v rámci jednotky práce může být řízen atributem MAXUMSGS příkazu ALTER QMGR na jiných platformách, s výjimkou operačního systému z/OS. Podrobné informace o těchto příkazech najdete v příručce Odkaz na MQSC *WebSphere MQ Script (MQSC) Command Reference*).

jednofázové potvrzení

Proces *jednofázové potvrzení* je takový, ve kterém může program potvrdit aktualizace fronty bez koordinace změn s ostatními správci prostředků.

Dvoufázové potvrzení

Proces *Dvoufázové potvrzení* je takový, v němž jsou aktualizace, které program provedl ve frontách produktu IBM WebSphere MQ , koordinovány s aktualizacemi jiných prostředků (například databázemi pod kontrolou DB2). Pod takovým procesem jsou aktualizace všech prostředků potvrzeny nebo zálohovány společně.

Pro práci s jednotkami práce poskytuje produkt IBM WebSphere MQ atribut *BackoutCount* . Tato hodnota se zvýší pokaždé, když se zazálohuje zpráva v rámci pracovní jednotky. Pokud zpráva opakovaně způsobí, že jednotka práce abnormálně skončí, hodnota *BackoutCount* nakonec překročí hodnotu zadanou *BackoutThreshold*. Tato hodnota je nastavena, když je definována fronta. V takové situaci může aplikace odebrat zprávu z pracovní jednotky a vložit ji do jiné fronty, jak je definováno v produktu *BackoutRequeueQName*. Když je zpráva přesunuta, může jednotka práce potvrdit.

Následující odkazy použijte k vyhledání dalších informací o potvrzení a zálohování jednotek práce:

- “Aspekty synchronizačních bodů v aplikacích produktu IBM WebSphere MQ” na stránce 311
- “Synchronizační body v IBM WebSphere MQ v systémech UNIX, Linux, and Windows” na stránce 312

Související pojmy

“Přehled rozhraní fronty zpráv” na stránce 187
Zde jsou uvedeny informace o komponentách rozhraní MQI (Message Queue Interface).

“Připojování k správci front a odpojování od něj” na stránce 198
Chcete-li používat programovací služby produktu WebSphere MQ , musí mít program připojení ke správci front. V této části se dozvíte, jak se připojit k správci front a odpojit je od správce front.

“Otevírání a zavírání objektů” na stránce 206
Tyto informace poskytují náhled na otevření a zavření objektů produktu WebSphere MQ .

“Vložení zpráv do fronty” na stránce 216
V této části se dozvíte, jak vložit zprávy do fronty.

“Získávání zpráv z fronty” na stránce 231
Tyto informace použijte k získání informací o získávání zpráv z fronty.

“Inquaning about a nastavení atributů objektu” na stránce 307
Atributy jsou vlastnosti, které definují charakteristiky objektu produktu WebSphere MQ .

“Spuštění aplikací produktu IBM WebSphere MQ pomocí spouštěčů” na stránce 316
Informace o spouštěčích a o tom, jak spustit aplikace IBM WebSphere MQ pomocí spouštěčů.

“Práce s MQI a klastry” na stránce 332
Existují speciální volby na voláních a návratových kódech, které se vztahují ke klastrování.

Aspekty synchronizačních bodů v aplikacích produktu IBM WebSphere MQ

Tyto informace použijte k seznámení se s použitím synchronizačních bodů v aplikacích produktu IBM WebSphere MQ .

Dvoufázové potvrzování je podporováno v rámci:

- WebSphere MQ for AIX
- WebSphere MQ for HP-UX
- WebSphere MQ for Linux
- WebSphere MQ pro Solaris
- WebSphere MQ for Windows
- CICS pro MVS/ESA 4.1
- CICS Transaction Server pro z/OS
- TXSeries
- IMS/ESA

- Další externí koordinátoři pomocí rozhraní X/Open XA

Jednofázové potvrzování je podporováno v rámci:

- WebSphere MQ na systémech UNIX
- WebSphere MQ for Windows

Poznámka: Další podrobnosti o externích rozhraních viz “Rozhraní pro externí správce synchronizačního bodu” na stránce 314a dokumentaci XA *CAE Specification Distributed Transaction Processing: The XA Specification*, publikované skupinou Open Group. Správci transakcí (například CICS, IMS, Encinaa Tuxedo) se mohou podílet na dvoufázovém potvrzování koordinovaném s jinými zotavitelné prostředky. To znamená, že funkce front poskytované produktem WebSphere MQ mohou být přeneseny do rozsahu pracovní jednotky, kterou spravuje správce transakcí.

Ukázky dodávané s produktem WebSphere MQ ukazují produkt WebSphere MQ koordinující databáze standardu XA. Další informace o těchto ukázkách naleznete v tématu “Ukázkové programy pro distribuované platformy” na stránce 93.

V aplikaci WebSphere MQ můžete v každé operaci vložení a volání určit, zda má být volání pod řízením synchronizačního bodu. Chcete-li provést operaci vložení pod řízením synchronizačního bodu, použijte hodnotu MQPMO_SYNCPOINT v poli *Options* struktury MQPMO, když voláte příkaz MQPUT. V případě operace get použijte hodnotu MQGMO_SYNCPOINT v poli *Options* struktury MQGMO. Pokud volbu nevyberete explicitně, závisí výchozí akce na platformě. Výchozí ovládací prvek syncpoint ne.

Je-li volání MQPUT1 vydáno s MQPMO_SYNCPOINT, změní se výchozí chování, takže je operace vložení dokončena asynchronně. To může způsobit změnu chování některých aplikací, které závisí na určitých polích ve strukturách MQOD a MQMD, které jsou vraceny, ale které nyní obsahují nedefinované hodnoty. Aplikace může určit MQPMO_SYNC_RESPONSE, aby se zajistilo, že je operace umístění provedena synchronně a že jsou dokončeny všechny odpovídající hodnoty polí.

Když aplikace obdrží kód příčiny MQRC_BACLED_OUT v odpovědi na synchronizační bod MQPUT nebo MQGET v rámci synchronizačního bodu, měla by aplikace za normálních okolností vrátit aktuální transakci pomocí operace MQBACK a poté v případě potřeby zkusit znovu celou transakci. Pokud aplikace přijme odpověď MQRC_BACKED_OUT v odpovědi na volání MQCMIT nebo MQDISC, není nutné volat MQBACK.

Při každém vrácení volání MQGET se zvýší hodnota pole *BackoutCount* struktury MQMD ovlivněné zprávy. Vysoká hodnota *BackoutCount* označuje zprávu, která byla opakovaně vrácena. To může naznačovat problém s touto zprávou, kterou byste měli vyšetřit. Podrobnosti viz BackoutCount , kde najdete podrobnosti o *BackoutCount*.

Pokud program vydá volání MQDISC, zatímco existují nepotvrzené požadavky, dojde k implicitnímu synchronizačnímu bodu. Pokud se program ukončí nestandardně, dojde k implicitnímu vrácení.

Změny atributů fronty (volání MQSET nebo pomocí příkazů) nejsou ovlivněny potvrzením nebo zálohováním jednotek práce.

Synchronizační body v IBM WebSphere MQ v systémech UNIX, Linux, and Windows

Podpora synchronizačních bodů funguje na dvou typech jednotek práce: lokální a globální.

Lokální jednotka práce je taková, ve které jsou jedinými aktualizovanými prostředky ty správce front produktu WebSphere MQ . Zde je koordinace synchronizačního bodu zajišťována samotným správcem front pomocí procedury jednofázového potvrzování.

Globální jednotka práce je taková, v níž jsou aktualizovány také prostředky náležící jiným správcům prostředků, jako jsou databáze. Produkt WebSphere MQ může koordinovat takové jednotky práce samotné. Mohou být také koordinovány externím řídícím řadičem, jako je jiný správce transakcí nebo IBM i prováděcí řadič.

Pro úplnou integritu použijte proceduru s dvoufázovým potvrzováním. Dvoufázové potvrzování mohou být poskytovány správci transakcí kompatibilními s XA a databázemi, jako jsou IBM' s TXSeries a UDB.

Produkty WebSphere MQ (kromě produktů WebSphere MQ for IBM i a WebSphere MQ for z/OS) mohou koordinovat globální jednotky práce pomocí procesu s dvoufázovým potvrzováním.

Místní jednotky práce

Jednotky práce, které zahrnují pouze správce front, se nazývají *lokální* jednotky práce. Koordinace synchronizačního bodu je poskytována samotným správcem front (interní koordinace) pomocí procesu jednofázového potvrzení.

Chcete-li spustit lokální jednotku práce, aplikace vydá požadavky MQGET, MQPUT nebo MQPUT1 s určením příslušné volby synchronizačního bodu. Jednotka práce je potvrzena pomocí MQCMIT nebo odvolána pomocí operace MQBACK. Pracovní jednotka je však také ukončena, když je přerušeno spojení mezi aplikací a správcem front úmyslně nebo neúmyslně.

Pokud dojde k odpojení aplikace (MQDISC) ze správce front, zatímco globální transakce koordinovaná produktem WebSphere MQ je stále aktivní, dojde k pokusu o potvrzení jednotky práce. Je-li však aplikace ukončena bez odpojení, jednotka práce je odvolána, protože se má za to, že se žádost ukončila abnormálně.

Globální pracovní jednotky

Použijte globální jednotky práce, pokud budete také muset zahrnout aktualizace prostředků náležících do jiných správců prostředků.

Zde může být koordinace interní nebo externí pro správce front:

Interní koordinace synchronizačního bodu

Koordinace globálních transakcí správce front není podporována produktem WebSphere MQ for IBM i nebo WebSphere MQ for z/OS. Toto není podporováno v prostředí klienta WebSphere MQ MQI.

Zde WebSphere MQ provádí koordinaci. Chcete-li spustit globální jednotku práce, aplikace vydá volání MQBEGIN.

Jako vstup do volání MQBEGIN je třeba zadat manipulátor připojení (*Hconn*) vrácený voláním MQCONN nebo MQCONNX. Tento manipulátor představuje připojení ke správci front produktu WebSphere MQ .

Aplikace vydá požadavky MQGET, MQPUT nebo MQPUT1 s určením příslušné volby synchronizačního bodu. To znamená, že můžete pomocí funkce MQBEGIN zahájit globální pracovní jednotku, která aktualizuje lokální prostředky, prostředky patřící jiným správcům prostředků, nebo obojí. Aktualizace prostředků náležejících k jiným správcům prostředků jsou prováděny pomocí rozhraní API daného správce prostředků. Rozhraní MQI však nelze použít k aktualizaci front, které patří k jiným správcům front. Před spuštěním dalších jednotek práce (lokální nebo globální) zadejte MQCMIT nebo MQBACK.

Globální pracovní jednotka je potvrzena pomocí MQCMIT, což iniciuje dvoufázové potvrzování všech správců prostředků zapojených do pracovní jednotky. Proces dvoufázového potvrzování se používá v případech, kdy jsou správci prostředků (například správci databází kompatibilní se standardem XA jako DB2, Oraclea Sybase) nejprve vyzváni k přípravě na potvrzení. Pouze v případě, že jsou všechny připraveny, jsou vyzvány k odevzdání. Pokud některý správce prostředků signalizuje, že jej nemůže potvrdit, bude každý z nich požádán o vrácení zpět. Případně můžete použít příkaz MQBACK k odvolání aktualizací všech správců prostředků.

Pokud se aplikace odpojí (MQDISC), zatímco je stále aktivní globální transakce, jednotka práce je potvrzena. Je-li však aplikace ukončena bez odpojení, jednotka práce je odvolána, protože se má za to, že se žádost ukončila abnormálně.

Výstup z MQBEGIN je kód dokončení a kód příčiny.

Při spuštění globální jednotky práce pomocí příkazu MQBEGIN jsou zahrnuty všechny externí správce prostředků, kteří byli konfigurováni s použitím správce front. Volání však spustí jednotku práce, ale dokončí se s varováním, pokud:

- Neexistují žádné zúčastněné správce prostředků (to znamená, že u správce front nebyli konfigurováni žádní správci prostředků)

, nebo

- Jeden nebo více správců prostředků není k dispozici.

V těchto případech musí jednotka práce zahrnovat aktualizace pouze na ty správce prostředků, kteří byli dostupní při spuštění jednotky práce.

Pokud jeden ze správců prostředků nemůže potvrdit své aktualizace, jsou správci prostředků vyzváni k odvolání jejich aktualizací a produkt MQCMIT je dokončen s varováním. Za neobvyklých okolností (obvykle zásah operátora) může volání MQCMIT selhat, pokud některé správce prostředků potvrdí své aktualizace, ale ostatní je vrátí zpět; práce je považována za dokončenou s výsledkem *smíšený* výsledek. Tyto výskyty jsou diagnostikovány v protokolu chyb správce front, aby bylo možné provést nápravnou akci.

MQCMIT globální transakce uspěje v případě, že všechny správci prostředků potvrdí své aktualizace.

Popis volání MQBEGIN viz MQBEGIN.

Koordinace externího synchronizačního bodu

K tomu dojde v případě, že byl vybrán jiný koordinátor syncpoint než produkt WebSphere MQ ; například CICS, Encinanebo Tuxedo.

V této situaci produkt WebSphere MQ v systému UNIX and Linux a produkt WebSphere MQ for Windows zaregistrují svůj zájem na výsledku transakce se koordinátorem synchronizačního bodu, aby mohli potvrdit nebo odvolat všechny nepotvrzené operace get nebo put, jak je požadováno. Koordinátor externího synchronizačního bodu určuje, zda jsou poskytnuty jednofázové a dvoufázové protokoly závazků.

Při použití externího koordinátora nelze vydat příkaz MQCMIT, MQBACK a MQBEGIN. Volání těchto funkcí se nezdaří s kódem příčiny MQRC_ENVIRONMENT_ERROR.

Způsob, jakým je externě koordinovaná jednotka práce spuštěna, závisí na programovacím rozhraní poskytovaném koordinátorem synchronizačního bodu. Je možné, že se požaduje explicitní volání. Je-li vyžadováno explicitní volání a zadáte-li volbu MQPMO_SYNCPOINT při spouštění jednotky práce, bude vrácen kód dokončení MQRC_SYNCPINT_NOT_AVAILABLE.

Rozsah jednotky práce je určen koordinátorem synchronizačního bodu. Stav připojení mezi aplikací a správcem front má vliv na úspěch nebo selhání volání MQI, které vyvolává problémy aplikace, nikoli stav jednotky práce. Aplikace může například odpojit a znovu připojit ke správci front během aktivní pracovní jednotky a provádět další operace MQGET a MQPUT uvnitř stejné jednotky práce. Tento stav je znám jako nevyřízené odpojení.

Můžete použít volání rozhraní API produktu WebSphere MQ v programech CICS , ať už se rozhodnete použít schopnosti XA systému CICS. Pokud nepoužíváte volbu XA, operace vložení a získávání zpráv do front a z nich nebudou spravovány v rámci CICS atomických jednotek práce. Jedním z důvodů pro výběr této metody je to, že celková konzistence jednotky práce není pro vás důležitá.

Je-li pro vás důležitá integrita vašich jednotek práce, musíte použít standard XA. Když používáte standard XA, systém CICS používá protokol dvoufázového potvrzování k zajištění toho, aby všechny prostředky v rámci jednotky práce byly aktualizovány společně.

Další informace o nastavení transakční podpory naleznete v příručce “Scénáře transakčního podpory” na stránce 39a také v dokumentaci TXSeries CICS , například *TXSeries for Multiplatforms CICS Administration Guide for Open Systems*.

Rozhraní pro externí správce synchronizačního bodu

WebSphere MQ v systémech UNIX and Linux , a WebSphere MQ for Windows podporují koordinaci transakcí externími správci synchronizačního bodu, kteří používají rozhraní XA sdružení X/Open.

Někteří správci transakcí XA (TXSeries) vyžadují, aby každý správce prostředků XA doddat svůj název. Jedná se o řetězec s názvem name ve struktuře přepínačů XA. Správce prostředků pro produkt WebSphere MQ na systémech UNIX, Linux a Windows se nazývá MQSeries_XA_RMI. Další podrobnosti o rozhraních XA najdete v dokumentaci XA *CAE Specification Distributed Transaction Processing: The XA Specification*(Specifikace XA) publikovaná skupinou Open Group.

V konfiguraci XA produkt WebSphere MQ v systémech UNIX, Linuxa Windows plní roli XA Resource Manager. Koordinátor synchronizačních bodů XA může spravovat sadu správců prostředků XA a synchronizovat potvrzení nebo vrácení transakcí ve Správci prostředků. Takto pracuje staticky zaregistrovaný správce prostředků:

1. Aplikace oznámí koordinátorovi synchronizačního bodu, že chce spustit transakci.
2. Koordinátor syncpoint vydá výzvu všem správcům prostředků, o kterých ví, že je upozorní na aktuální transakci.
3. Aplikace vydá výzvu k aktualizaci prostředků spravovaných správci prostředků přidruženým k aktuální transakci.
4. Aplikace požaduje, aby koordinátor syncpoint buď potvrdil, nebo odvolal transakci.
5. Koordinátor synchronizačního bodu vydá volání každému správci prostředků pomocí protokolů s dvoufázovým potvrzováním k dokončení transakce, jak je požadováno.

Specifikace XA vyžaduje, aby každý Resource Manager poskytoval strukturu s názvem *Přepínač XA*. Tato struktura deklaruje schopnosti správce prostředků Resource Managera funkce, které mají být volány koordinátorem synchronizačního bodu.

K dispozici jsou dvě verze této struktury:

MQRMIXASwitch	Správa statických prostředků XA
MQRMIXASwitchDynamic	Správa dynamických prostředků XA

Seznam knihoven obsahujících tuto strukturu naleznete v tématu “Struktura přepínače IBM WebSphere MQ XA” na stránce 67.

Metoda, která musí být použita k propojení s koordinátorem synchronizačního bodu XA, je definována koordinátorem; v dokumentaci poskytnuté tímto koordinátorem zjistěte, jak povolit produkt WebSphere MQ ke spolupráci s koordinátorem synchronizačního bodu XA.

Struktura *xa_info* , která se předává všem voláním *xa_open* koordinátorem synchronizačního bodu, může být název správce front, který má být spravován. Jedná se o stejný formulář jako název správce front předaný MQCONN nebo MQCONNX a může být prázdný, pokud má být použit výchozí správce front. Avšak můžete použít dva extra parametry TPM a AXLIB

Produkt TPM umožňuje zadat do produktu WebSphere MQ název správce transakcí, například CICS. Systém AXLIB umožňuje určit skutečný název knihovny ve správci transakcí, ve kterém jsou umístěny vstupní body XA AX.

Pokud použijete některý z těchto parametrů nebo jiného než výchozího správce front, musíte zadat název správce front pomocí parametru QMNAME. Další informace naleznete v části Parametry CHANNEL, TRPTYPE, CONNAME a QMNAME řetězce xa_open.

Omezení

1. Globální jednotky práce nejsou povoleny se sdíleným Hconn (jak je popsáno v “Sdílená připojení (nezávislá na podprocesech) s MQCONNX” na stránce 204.
2. Na systémech Windows jsou všechny funkce deklarované v přepínači XA deklarovány jako funkce _cdecl.
3. Koordinátor externího synchronizačního bodu může v daném okamžiku spravovat pouze jednoho správce front. Důvodem je skutečnost, že koordinátor má efektivní připojení ke každému správci front, a proto podléhá pravidlu, že v daném okamžiku je povoleno pouze jedno připojení.

   Poznámka: Poznámka: Aplikace klienta JMS (aplikace CLIENT JEE) spuštěná na serveru JEE nemá toto omezení, takže jedna transakce spravovaná serverem JEE může zkoordinovat více správců front v rámci stejné transakce. Nicméně aplikace serveru JMS, která běží v režimu vazeb, je stále předmětem pravidla, že v daném okamžiku je povoleno pouze jedno připojení.
4. Všechny aplikace, které jsou spuštěny pomocí koordinátora synchronizačního bodu, se mohou připojit pouze ke správci front, který je spravován koordinátorem, protože jsou již k tomuto správci front

účinně připojeni. Musí vydat příkaz MQCONN nebo MQCONNX k získání manipulátoru připojení a před ukončením operace musí vydat MQDISC. Případně mohou použít uživatelskou proceduru UE014015 pro TXSeries CICS.

Spuštění aplikací produktu IBM WebSphere MQ pomocí spouštěčů

Informace o spouštěčích a o tom, jak spustit aplikace IBM WebSphere MQ pomocí spouštěčů.

Některé aplikace produktu WebSphere MQ , které obsluhují fronty, jsou spuštěny nepřetržitě, takže jsou vždy k dispozici pro načtení zpráv, které přicházejí do front. Pokud však počet příchozích zpráv doručené do front není možný, může to být nepředvídatelné. V takovém případě mohou aplikace spotřebovávat systémové prostředky i v případě, že nejsou k dispozici žádné zprávy k načtení.

Produkt WebSphere MQ poskytuje službu, která umožňuje automatické spuštění aplikace v případě, že jsou k dispozici zprávy k načtení. Tato služba je známá jako *spouštějící*.

Informace o spouštění kanálů naleznete v části Spouštěcí kanály .

Co se spouští?

Správce front definuje určité podmínky jako *spouštěcí události*.

Je-li pro frontu povoleno spouštění a dojde k události spouštěče, správce front odešle *zprávu spouštěče* do fronty nazývané *inicializační fronta*. Přítomnost zprávy spouštěče v inicializační frontě indikuje, že došlo k události spouštěče.

Zprávy spouštěče generované správcem front nejsou trvalé. Tím se sníží protokolování (výsledkem je zlepšení výkonu) a minimalizace duplikátů během restartu, takže se zlepší doba restartování.

Program, který zpracovává inicializační frontu, se nazývá *aplikace monitoru spouštěčů*a její funkcí je číst zprávu spouštěče a provést odpovídající akci na základě informací obsažených ve zprávě spouštěče. Obvykle má tato akce spustit některou jinou aplikaci pro zpracování fronty, která generovala zprávu spouštěče. Z pohledu správce front není k dispozici žádná speciální informace o aplikaci pro monitor spouštěčů; jedná se pouze o jinou aplikaci, která čte zprávy z fronty (inicializační fronta).

Pokud je pro frontu povoleno spouštění, můžete k němu přidružit *objekt definice procesu* přidružený k této frontě. Tento objekt obsahuje informace o aplikaci, která zpracovává zprávu, která způsobila událost spouštěče. Je-li vytvořen objekt definice procesu, správce front extrahuje tyto informace a umístí jej do zprávy spouštěče, kterou použije aplikace pro monitor spouštěčů. Název definice procesu přidružený ke frontě je dán atributem local-queue produktu *ProcessName* . Každá fronta může určovat jinou definici procesu a několik front může sdílet stejnou.

Pokud chcete spustit spuštění kanálu, není nutné definovat objekt definice procesu. Místo toho se použije definice přenosové fronty.

Spouštěcí impuls je podporován klienty WebSphere MQ spuštěnými v následujících prostředích:

- Systémy UNIX and Linux
- Systémy Windows

Aplikace běžící v prostředí klienta je stejná jako aplikace spuštěná v úplném prostředí produktu WebSphere MQ , kromě toho, že ji propojíte s knihovnami klienta. Avšak spouštěcí monitor a aplikace, která má být spuštěny, musí být ve stejném prostředí.

Spouštění zahrnuje:

Fronta aplikací
Fronta aplikací je lokální fronta, která, když má spouštěcí impuls nastavený a jsou-li splněny podmínky, vyžaduje, aby byly zapsány zprávy spouštěče.

Definice procesu
K frontě aplikací může být přidružen *objekt definice procesu* , který obsahuje podrobné informace o aplikaci, která získá zprávy z aplikační fronty. (Viz Atributy pro definice procesu pro seznam atributů.)

Nezapomeňte, že pokud chcete, aby spouštěč spouštěl kanál, není třeba definovat objekt definice procesu.

Přenosová fronta
Chcete-li spustit kanál, potřebujete přenosovou frontu, chcete-li spustit spouštěč.

Pro přenosovou frontu v systémech AIX, HP-UX, IBM i, Solaris, z/OSnebo Windows může atribut *TriggerData* přenosové fronty určovat název kanálu, který má být spuštěn. Tímto způsobem lze nahradit definici procesu pro spouštěcí kanály, ale používá se pouze v případě, že definice procesu není vytvořena.

událost spouštěče
Událost spouštěče je událost, která způsobí generování zprávy spouštěče správcem front. Jedná se obvykle o zprávu přicházející do aplikační fronty, ale může k ní dojít také v jiných časech (viz “Podmínky pro událost spouštěče” na stránce 321). Produkt WebSphere MQ má řadu voleb, které vám umožňují řídit podmínky, které způsobují událost triggeru (viz “Řízení událostí spouštěče” na stránce 325).

zpráva spouštěče
Správce front vytvoří *spouštěcí zprávu* , pokud rozpozná událost triggeru (viz “Podmínky pro událost spouštěče” na stránce 321). Zkopíruje do zprávy spouštěče informace o aplikaci, která má být spuštěna. Tyto informace pocházejí z fronty aplikace a objektu definice procesu přidruženého k aplikační frontě. Zprávy spouštěče mají pevný formát (viz “Formát zpráv spouštěče” na stránce 331).

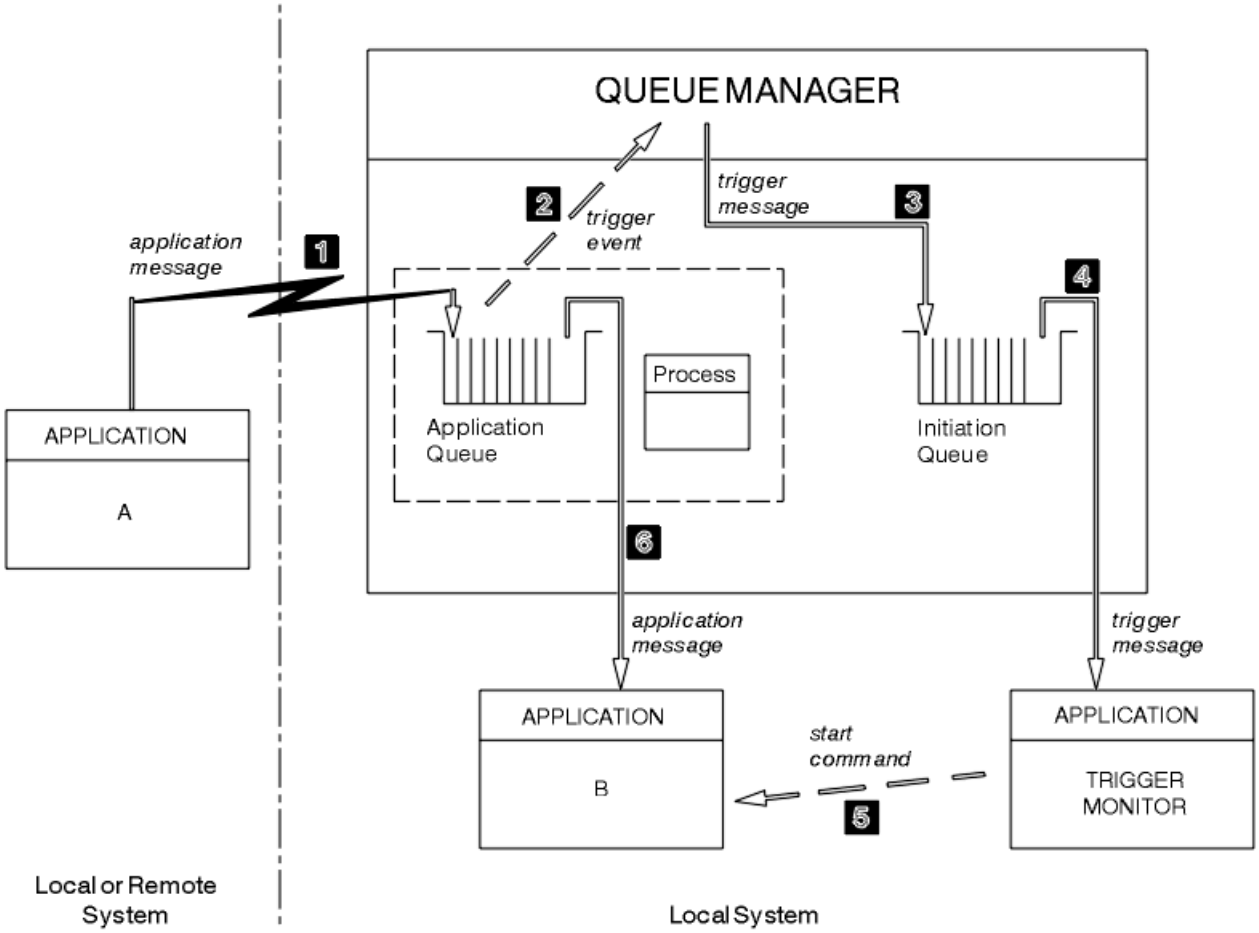
Inicializační fronta
Inicializační fronta je lokální fronta, do níž správce front vkládá zprávy spouštěče. Všimněte si, že inicializační fronta nemůže být alias fronta nebo modelová fronta. Správce front může vlastnit více než jednu inicializační frontu a každý z nich je přidružen k jedné nebo více frontám aplikace. Sdílená fronta, lokální fronta přístupná pro správce front ve skupině sdílení front, může být inicializační frontou v produktu WebSphere MQ pro systém z/OS.

monitor spouštěčů
Monitor spouštěčů je nepřetržitý spuštěný program, který obsluhuje jednu nebo více inicializačních front. Když do inicializační fronty přijde zpráva spouštěče, načte tuto zprávu monitor spouštěčů. Monitor spouštěčů používá informace ve zprávě spouštěče. Vydává příkaz ke spuštění aplikace, která má načíst zprávy přicházející do fronty aplikací a předávají ji informace obsažené v záhlaví zprávy spouštěče, která obsahuje název fronty aplikací.

Na všech platformách je odpovědný za spouštění kanálů speciální monitor spouštěčů, který je znám jako iniciátor kanálu. V systému z/OSje inicializátor kanálu zpravidla spuštěn ručně nebo může být proveden automaticky při spuštění správce front změnou hodnoty CSQINP2 ve spouštěcím skriptu JCL správce front. Na jiných platformách se při spuštění správce front automaticky spustí nebo je možné jej spustit ručně pomocí příkazu runmqchi.

(Další informace viz “Inicializační fronta zpracovává monitory spouštěčů” na stránce 328.)

Chcete-li porozumět tomu, jak spouštění funguje, zvažte produkt Obrázek 68 na stránce 318, který je příkladem typu spouštěče FIRST (MQTT_FIRST).



Obrázek 68. Tok zpráv aplikace a spouštěče

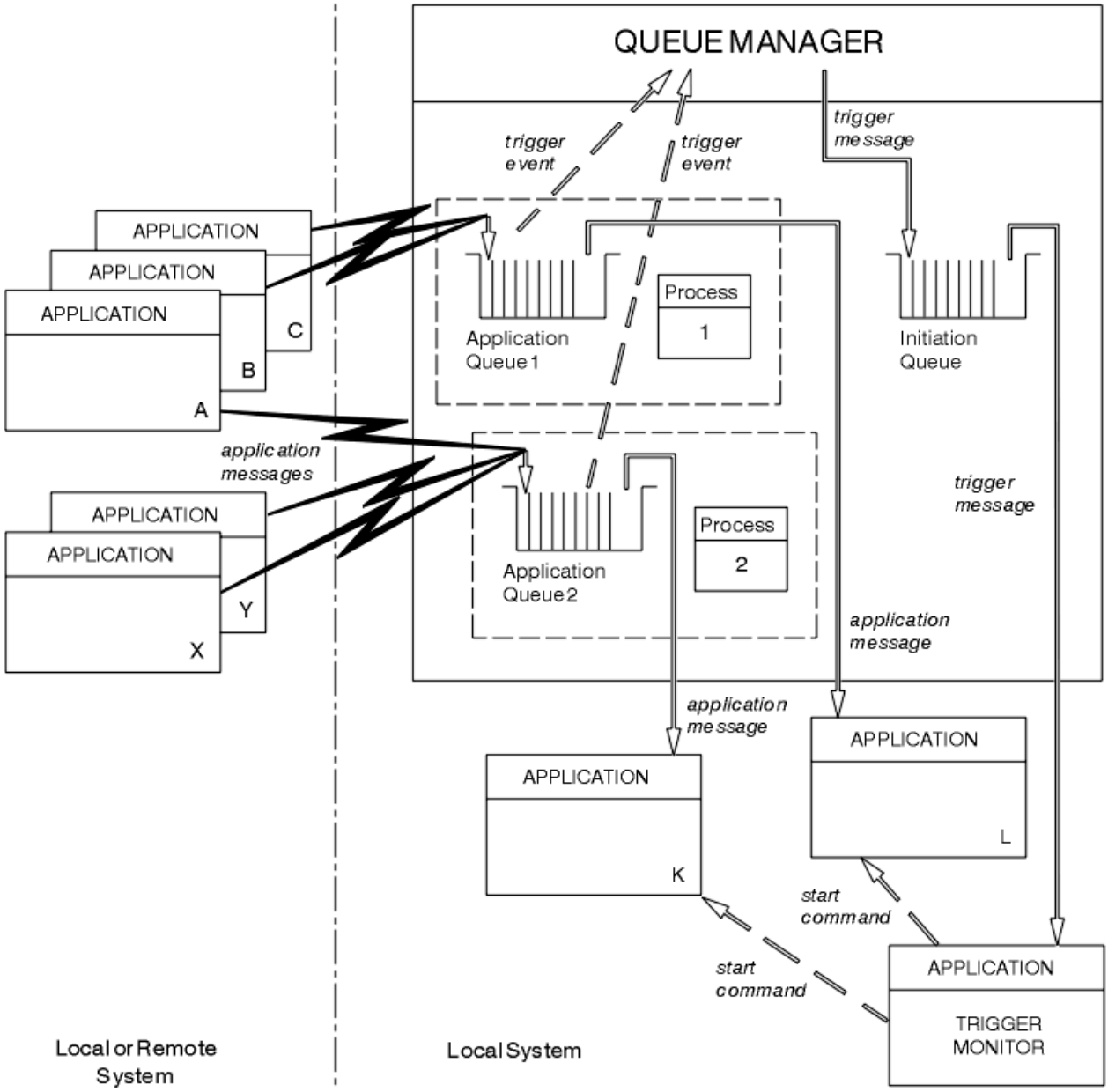
V produktu Obrázek 68 na stránce 318je posloupnost událostí následující:

1. Aplikace A, která může být buď lokální, nebo vzdálená vzhledem ke správci front, vloží zprávu do fronty aplikací. Žádná aplikace nemá tuto frontu otevřenou pro vstup. Tato skutečnost je však relevantní pouze pro typ spouštěče FIRST a DEPTH.
2. Správce front zkontroluje, zda jsou splněny podmínky, za kterých musí generovat událost spouštěče. Jsou, a je generována událost spouštěče. Informace uchovávané v přidruženém objektu definice procesu se používají při vytváření zprávy spouštěče.
3. Správce front vytvoří zprávu spouštěče a vloží ji do inicializační fronty přidružené k této frontě aplikací, ale pouze v případě, že má aplikace (monitor spouštěčů) otevřenou vstupní frontu pro vstup.
4. Monitor spouštěčů načte zprávu spouštěče z inicializační fronty.
5. Monitor spouštěčů vydá příkaz pro spuštění aplikace B (serverová aplikace).
6. Aplikace B otevře aplikační frontu a načte zprávu.

Poznámka:

1. Je-li fronta aplikací otevřena pro vstup, libovolným programem a má spouštěcí sadu pro FIRST nebo DEPTH, nedojde k žádné události spouštěče, protože fronta je již obsluhována.
2. Není-li inicializační fronta otevřena pro vstup, správce front negeneruje žádné zprávy spouštěče. Bude čekat, dokud aplikace neotevře inicializační frontu pro vstup.
3. Při použití spouštěče pro kanály použijte typ spouštěče FIRST nebo DEPTH.
4. Spuštěné aplikace jsou spuštěny pod ID uživatele a skupinou uživatele, který spustil monitor spouštěčů, uživatele produktu CICS nebo uživatele, který spustil správce front.

Dosud byl vztah mezi frontami v rámci spuštění pouze na jednom z nich. Zvažte Obrázek 69 na stránce 319.



Obrázek 69. Vztah front v rámci spouštěče

Fronta aplikace má přidružený objekt definice procesu, který uchovává podrobnosti o aplikaci, která bude zpracovávat tuto zprávu. Správce front umístí informace do zprávy spouštěče tak, aby byla nutná pouze jedna inicializační fronta. Monitor spouštěčů extrahuje tyto informace ze zprávy spouštěče a spustí příslušnou aplikaci, aby se mohla vypořádat se zprávou na každé frontě aplikací.

Pamatujte na to, že pokud chcete spustit spuštění kanálu, nemusíte definovat objekt definice procesu. Definice přenosové fronty může určit, který kanál se má spustit.

Následující odkazy použijte k vyhledání dalších informací o spouštění aplikací produktu WebSphere MQ pomocí spouštěčů:

- “Nezbytné předpoklady pro spuštění” na stránce 320
- “Podmínky pro událost spouštěče” na stránce 321
- “Řízení událostí spouštěče” na stránce 325

- “Návrh aplikace, která používá spuštěné fronty” na stránce 327
- “Inicializační fronta zpracovává monitory spouštěčů” na stránce 328
- “Vlastnosti zpráv spouštěče” na stránce 331
- “Když spouštění nefunguje” na stránce 332

Související pojmy

“Přehled rozhraní fronty zpráv” na stránce 187
Zde jsou uvedeny informace o komponentách rozhraní MQI (Message Queue Interface).

“Připojování k správci front a odpojování od něj” na stránce 198
Chcete-li používat programovací služby produktu WebSphere MQ , musí mít program připojení ke správci front. V této části se dozvíte, jak se připojit k správci front a odpojit je od správce front.

“Otevírání a zavírání objektů” na stránce 206
Tyto informace poskytují náhled na otevření a zavření objektů produktu WebSphere MQ .

“Vložení zpráv do fronty” na stránce 216
V této části se dozvíte, jak vložit zprávy do fronty.

“Získávání zpráv z fronty” na stránce 231
Tyto informace použijte k získání informací o získávání zpráv z fronty.

“Inquaning about a nastavení atributů objektu” na stránce 307
Atributy jsou vlastnosti, které definují charakteristiky objektu produktu WebSphere MQ .

“Potvrzení a zálohování jednotek práce” na stránce 310
Tyto informace popisují, jak potvrdit a zazálohovat všechny zotavitelné operace get a put, které se vyskytly v jednotce práce.

“Práce s MQI a klastry” na stránce 332
Existují speciální volby na voláních a návratových kódech, které se vztahují ke klastrování.

Nezbytné předpoklady pro spuštění

Tyto informace použijte k seznámení se s kroky, které je třeba provést před použitím spouštěče.

Před tím, než bude vaše aplikace moci využívat spouštění, postupujte takto:

1. Proveďte jednu z následujících akcí:
   a. Vytvořte inicializační frontu pro frontu aplikací. Příklad:

```
DEFINE QLOCAL (initiation.queue) REPLACE      +
       LIKE (SYSTEM.DEFAULT.INITIATION.QUEUE)      +
       DESCR ('initiation queue description')
```

   , nebo

   b. Určete název lokální fronty, která existuje a může ji použít vaše aplikace (obvykle se jedná o název SYSTEM.DEFAULT.INITIATION.QUEUE nebo, pokud spouštíte kanály se spouštěči, SYSTEM.CHANNEL.INITQ) a zadejte její název do pole *InitiationQName* ve frontě aplikací.

2. Přidružte inicializační frontu k frontě aplikací. Správce front může vlastnit více než jednu inicializační frontu. Možná budete chtít, aby některé z vašich aplikačních front byly obsluhovány různými programy, v tom případě můžete použít jednu inicializační frontu pro každý obsluhující program, i když to nemusíte. Zde je příklad, jak vytvořit aplikační frontu:

```
DEFINE QLOCAL (application.queue) REPLACE   +
LIKE (SYSTEM.DEFAULT.LOCAL.QUEUE)           +
DESCR ('appl queue description')    +
INITQ ('initiation.queue')          +
PROCESS ('process.name')            +
TRIGGER                                     +
TRIGTYPE (FIRST)
```

3. Pokud spustíte aplikaci, vytvořte objekt definice procesu, který bude obsahovat informace vztahující se k aplikaci, která má sloužit ke zpracování vaší aplikační fronty. Chcete-li například spustit příkaz CICS payroll transakce s názvem PAYR, spusťte:

```
DEFINE PROCESS (process.name) +
       REPLACE +
       DESCR ('process description') +
       APPLICID ('PAYR') +
       APPLTYPE (CICS) +
       USERDATA ('Payroll data')
```

   Když správce front vytvoří zprávu spouštěče, okopíruje informace z atributů objektu definice procesu do zprávy spouštěče.

Platforma	Vytvoření objektu definice procesu
Systémy UNIX, Linuxa Windows	Použijte atribut DEFINE PROCESS nebo použijte SYSTEM.DEFAULT.PROCESS a úprava pomocí příkazu ALTER PROCESS

4. Volitelné: Vytvořte definici přenosové fronty a použijte prázdné místo pro atribut *ProcessName* .

   Atribut *TrigData* může obsahovat název kanálu, který má být spuštěn, nebo jej lze ponechat prázdný. S výjimkou IBM WebSphere MQ for z/OS, pokud je ponechán prázdný, prohledává inicializátor kanálu soubory s definicemi kanálů, dokud nenajde kanál, který je přidružen k uvedené přenosové frontě. Když správce front vytvoří zprávu spouštěče, zkopíruje informace z atributu *TrigData* definice přenosové fronty do zprávy spouštěče.

5. Pokud jste vytvořili objekt definice procesu k určení vlastností aplikace, která slouží ke zpracování vaší aplikační fronty, přidružte objekt procesu k frontě aplikací tím, že ji pojmenujete v atributu *ProcessName* fronty.

Platforma	Použít příkazy
Systémy UNIX, Linuxa Windows	POZMĚNIT QLOCAL

6. Spouštět instance spouštěče , které mají sloužit pro inicializační fronty, které jste definovali. Další informace viz “Inicializační fronta zpracovává monitory spouštěčů” na stránce 328.

Chcete-li si být vědomi jakýchkoli nedoručených zpráv spouštěče, ujistěte se, že má správce front definovanou frontu nedoručených zpráv (nedoručená zpráva). Do pole správce front produktu *DeadLetterQName* zadejte název fronty.

Poté můžete nastavit podmínky spouštěče, které vyžadujete, s použitím atributů objektu fronty, který definuje vaši frontu aplikací. Další informace viz “Řízení událostí spouštěče” na stránce 325.

Podmínky pro událost spouštěče

Odkazy na sdílené fronty v tomto tématu znamenají sdílené fronty ve skupině sdílení front, které jsou k dispozici pouze v produktu WebSphere MQ pro systém z/OS.

Správce front vytvoří zprávu spouštěče, jsou-li splněny následující podmínky:

1. Zpráva je ve frontě *vložena* do fronty.
2. Zpráva má prioritu větší nebo rovnou prahové hodnotě triggeru prahové hodnoty fronty. Tato priorita je nastavena v atributu lokální fronty *TriggerMsgPriority* ; pokud je nastavena na nulu, libovolná zpráva se kvalifikuje.
3. Počet zpráv ve frontě s prioritou vyšší nebo rovnou *TriggerMsgPriority* byl dříve, v závislosti na *TriggerType*:
   - Nula (pro typ spouštěče MQTT_FIRST)

- Libovolné číslo (pro typ spouštěče MQTT_EVERY)
- *TriggerDepth* minus 1 (pro typ spouštěče MQTT_DEPTH)

Poznámka:

a. U nesdílených lokálních front počítá správce front jak potvrzené, tak nepotvrzené zprávy, když vyhodnocuje, zda existují podmínky pro událost spouštěče. Proto může být aplikace spuštěna, když nejsou k dispozici žádné zprávy k načtení, protože zprávy ve frontě nebyly potvrzeny. V této situaci zvažte použití volby čekání s vhodným prostorem *WaitInterval*, takže aplikace čeká na příchod svých zpráv.

b. U lokálních sdílených front počítá správce front pouze potvrzené zprávy.

4. Pro spuštění typu FIRST nebo DEPTH žádný program nemá otevřenou frontu aplikace pro odstranění zpráv (tj. lokální atribut fronty *OpenInputCount* je nula).

Poznámka:

a. Pro sdílené fronty platí speciální podmínky, pokud více správců front má spuštěno monitorování spuštěných proti frontě. V této situaci, pokud má jeden nebo více správců front otevřenou frontu pro sdílený vstup, jsou kritéria spouštěcího impulsu na ostatních správcích front považována za *TriggerType* MQTT_FIRST a *TriggerMsgPriority* nula. Když všechny správce front zavřou frontu pro vstup, obnoví se podmínky spouštěče na podmínky uvedené v definici fronty.

   Ukázkový scénář, který je ovlivněn touto podmínkou, je více správců front QM1, QM2a QM3 s monitorem spouštěčů spuštěným pro aplikační frontu A. Zpráva dorazí na A splňující podmínky pro spuštění a v inicializační frontě se vygeneruje zpráva spouštěče. Monitor spouštěčů na systému QM1 získá zprávu spouštěče a spustí aplikaci. Spuštěná aplikace otevře aplikační frontu pro sdílený vstup. Od tohoto bodu za spouštěcí podmínky pro frontu aplikací A se vyhodnotí jako *TriggerType* MQTT_FIRST a *TriggerMsgPriority* na správci front QM2 a QM3, dokud QM1 uzavře frontu aplikací.

b. Pro sdílené fronty je tato podmínka použita pro každého správce front. To znamená, že *OpenInputCount* správce front pro frontu musí být nula pro zprávu spouštěče, která má být generována pro danou frontu daným správcem front. Pokud však některý správce front ve skupině sdílení front má otevřenou frontu s použitím volby MQOO_INPUT_EXCLUSIVE, nebude pro danou frontu vygenerována žádná zpráva spouštěče pro danou frontu v rámci skupiny sdílení front.

   Změna způsobu vyhodnocení podmínek spouštěče se vyskytne, když spuštěná aplikace otevře frontu pro vstup. Ve scénářích, kde je spuštěný pouze jeden monitor spouštěčů, mohou mít jiné aplikace stejný účinek, protože podobně otevírají vstupní frontu pro vstup. Nezáleží na tom, zda byla aplikační fronta otevřena aplikací spuštěnou monitorem spouštěčů, nebo jinou aplikací; jedná se o skutečnost, že je fronta otevřena pro vstup na jiném správci front, který způsobuje změnu kritérií spouštěče.

5. Pokud je v produktu WebSphere MQ for z/OSfronta aplikací jedna s atributem *Usage* MQUS_NORMAL, získaní požadavky na tuto frontu (to znamená, že atribut fronty *InhibitGet* je MQQA_GETALLOWED). Je-li fronta aplikací spuštěna také s atributem *Usage* MQUS_XMITQ, nebudou mít k dispozici žádné požadavky, pro které nejsou blokovány žádné požadavky.
6. Proveďte jednu z následujících akcí:
   - Atribut lokální fronty *ProcessName* pro danou frontu není prázdný a byl vytvořen objekt definice procesu identifikovaný tímto atributem, nebo
   - Atribut lokální fronty *ProcessName* pro danou frontu je prázdný, ale fronta je přenosová fronta. Protože je definice procesu volitelná, může atribut *TriggerData* obsahovat také název kanálu, který má být spuštěn. V takovém případě bude zpráva spouštěče obsahovat atributy s následujícími hodnotami:
     - *QName*: název fronty
     - *ProcessName*: mezery
     - *TriggerData*: data spouštěče
     - *ApplType*: MQAT_UNKNOWN

- *ApplId*: mezery
- *EnvData*: mezery
- *UserData*: mezery

7. Inicializační fronta byla vytvořena a byla zadána v atributu lokální fronty *InitiationQName* . Dále:
   - Požadavky na získání nejsou blokovány pro inicializační frontu (to znamená, že atribut fronty *InhibitGet* je MQQA_GETALLOWED).
   - Požadavky PUT nesmí být blokovány pro inicializační frontu (to znamená, že atribut fronty *InhibitPut* musí být MQQA_PUT_ALLOWED).
   - Atribut *Usage* inicializační fronty musí být MQUS_NORMAL.
   - V prostředích, ve kterých jsou podporovány dynamické fronty, nesmí být inicializační fronta dynamická fronta, která byla označena jako logicky odstraněná.
8. Monitor spouštěčů má v současné době inicializační frontu otevřenou pro odstranění zpráv (to znamená, že lokální atribut fronty *OpenInputCount* je větší než nula).
9. Ovládací prvek spouštěče (atribut lokální fronty*TriggerControl*) pro frontu aplikací je nastaven na hodnotu MQTC_ON. Chcete-li to provést, nastavte atribut *trigger* při definování fronty nebo použijte příkaz ALTER QLOCAL.
10. Typ spouštěče (lokální atribut fronty*TriggerType*) není MQTT_NONE.

    Pokud jsou splněné všechny požadované podmínky a zpráva, která způsobila podmínku spouštěče, je vložena jako část pracovní jednotky, zpráva spouštěcího impulsu se nestane dostupnou pro načtení aplikací monitoru spouštěčů, dokud nebude dokončena jednotka práce, zda je jednotka práce potvrzena, nebo, pro typ triggeru MQTT_FIRST nebo MQTT_DEPTH, zálohováno.
11. Vhodná zpráva se umístí do fronty pro *TriggerType* MQTT_FIRST nebo MQTT_DEPTH a do fronty:
    - Nebyl dříve prázdný (MQTT_FIRST) nebo
    - Měl *TriggerDepth* nebo více zpráv (MQTT_DEPTH)

    a podmínky “2” na stránce 321 až “10” na stránce 323 (kromě produktu “3” na stránce 321) jsou splněny, pokud v případě MQTT_FIRST uplynul dostatečný interval (atribut správce front*TriggerInterval*) od doby, kdy byla pro tuto frontu zapsána poslední zpráva spouštěče.

    Toto je povolení pro server fronty, který se ukončí před zpracováním všech zpráv ve frontě. Účelem intervalu spouštěče je snížit počet vygenerovaných duplicitních zpráv spouštěče.

    Poznámka: Pokud zastavíte a znovu spustíte správce front, je resetován *časovač* *TriggerInterval* . Je zde malé okno, v jehož průběhu je možné vytvořit dvě zprávy spouštěče. Okno existuje, když je atribut spouštěče fronty nastaven tak, aby byl povolen ve stejnou dobu jako zpráva přijde a fronta nebyla dříve prázdná (MQTT_FIRST) nebo měla *TriggerDepth* nebo více zpráv (MQTT_DEPTH).
12. Jediná aplikace obsluhující frontu nabízí volání MQCLOSE, pro *TriggerType* MQTT_FIRST nebo MQTT_DEPTH, a je zde alespoň:
    - jeden (MQTT_FIRST) nebo
    - *TriggerDepth* (MQTT_DEPTH)

    jsou také zprávy ve frontě s dostatečnou prioritou (podmínka “2” na stránce 321) a podmínky “6” na stránce 322 až “10” na stránce 323 jsou také splněny.

    Toto je povolení pro server fronty, který vydává volání MQGET, vyhledá frontu prázdnou, a tak skončí; avšak v intervalu mezi voláními MQGET a MQCLOSE je doručena jedna nebo více zpráv.

    Poznámka:

    a. Pokud program obsluhující aplikační frontu nenačte všechny zprávy, může to způsobit zavřenou smyčku. Pokaždé, když program zavře frontu, vytvoří správce front další spouštěcí zprávu, která způsobí, že monitor spouštěčů znovu spustí program serveru.

b. Pokud program obsluhující frontu aplikací zálohuje svůj požadavek na získání (nebo pokud se program ukončí) před zavřením fronty, stane se totéž. Pokud však program frontu uzavře předtím, než dojde k získání požadavku na získání, a fronta je jinak prázdná, nebude vytvořena žádná zpráva spouštěče.

c. Chcete-li zabránit výskytu takové smyčky, použijte pole *BackoutCount* MQMD k detekci zpráv, které jsou opakovaně vráceny. Další informace viz "Zprávy, které jsou zálohovány" na stránce 36.

13. Pomocí MQSET nebo příkazu jsou splněny následující podmínky:

a. • *TriggerControl* se změní na MQTC_ON, nebo

• *TriggerControl* je již MQTC_ON a hodnota *TriggerType*, *TriggerMsgPriority*nebo *TriggerDepth* (je-li relevantní) se změní,

a je zde alespoň:

• jeden (MQTT_FIRST nebo MQTT_EVERY), nebo

• *TriggerDepth* (MQTT_DEPTH)

jsou také zprávy ve frontě s dostatečnou prioritou (podmínka "2" na stránce 321) a podmínky "4" na stránce 322 až "10" na stránce 323 (kromě "8" na stránce 323) jsou také splněny.

To umožňuje aplikaci nebo operátorovi, který mění spouštěcí kritéria, když již jsou splněny podmínky pro výskyt spouštěče.

b. Atribut fronty *InhibitPut* inicializační fronty se změní z hodnoty MQQA_PUT_INHIBITED na MQQA_PUT_ALLOWED a je nejméně:

• jeden (MQTT_FIRST nebo MQTT_EVERY), nebo

• *TriggerDepth* (MQTT_DEPTH)

zprávy s dostatečnou prioritou (podmínka "2" na stránce 321) ve všech frontách, pro které se jedná o inicializační frontu, a podmínky "4" na stránce 322 až "10" na stránce 323 jsou také splněny. (Jedna zpráva spouštěče je generována pro každou takovou frontu splňující podmínky.)

To umožňuje, aby se zprávy spouštěče negenerovaly kvůli podmínce MQQA_PUT_INHIBITED na inicializační frontě, ale tento stav se nyní změnil.

c. Atribut fronty produktu *InhibitGet* fronty aplikací se změní z hodnoty MQQA_GET_INHIBITED na MQQA_GET_ALLOWED a je alespoň:

• jeden (MQTT_FIRST nebo MQTT_EVERY), nebo

• *TriggerDepth* (MQTT_DEPTH)

zprávy s dostatečnou prioritou (podmínka "2" na stránce 321) ve frontě a podmínky "4" na stránce 322 až "10" na stránce 323, kromě "5" na stránce 322, jsou také splněny.

To umožňuje, aby aplikace byly spuštěny pouze tehdy, když mohou načítat zprávy z aplikační fronty.

d. Aplikace monitoru spouštěčů vydá volání MQOPEN pro vstup z inicializační fronty a je zde alespoň:

• jeden (MQTT_FIRST nebo MQTT_EVERY), nebo

• *TriggerDepth* (MQTT_DEPTH)

zprávy s dostatečnou prioritou (podmínka "2" na stránce 321) v jakékoli z aplikačních front, pro které se jedná o inicializační frontu, a podmínky "4" na stránce 322 až "10" na stránce 323 (kromě "8" na stránce 323) jsou také splněny, a žádná jiná aplikace nemá otevřenou frontu inicializace pro vstup (jedna zpráva spouštěče je generována pro každou takovou frontu splňující podmínky).

To znamená povolit zprávy přicházející do front, zatímco monitor spouštěčů není spuštěný, a pro restartování správce front a zprávy triggeru (které jsou přechodné), které se ztratí.

14. MSGDLVSQ je nastavena správně. Nastavíte-li parametr MSGDLVSQ=FIFO, budou zprávy doručovány do fronty nejprve v První odchozí zprávě. Priorita zprávy je ignorována a výchozí priorita fronty je přiřazena ke zprávě. Pokud je parametr *TriggerMsgPriority* nastaven na vyšší hodnotu, než je

výchozí priorita fronty, nespustí se žádné zprávy. Je-li parametr *TriggerMsgPriority* nastaven na hodnotu rovnou nebo nižší než výchozí priorita fronty, spustí se spuštění typu FIRST, EVERY a DEPTH. Informace o těchto typech naleznete v popisu pole *TriggerType* v části "Řízení událostí spouštěče" na stránce 325.

Pokud nastavíte MSGDLVSQ=PRIORITY a priorita zprávy se rovná nebo je větší než pole *TriggerMsgPriority* , zprávy *count* se vztahují pouze k události triggeru. V takovém případě se spustí spuštění typu FIRST, EVERY a DEPTH. Zadáte-li například 100 zpráv nižší priority než *TriggerMsgPriority*, efektivní hloubka fronty pro účely spuštění je stále nula. Pokud poté vložíte další zprávu do fronty, ale tentokrát je priorita větší nebo rovna hodnotě *TriggerMsgPriority*, bude se efektivní hloubka fronty zvětšovat od nuly do jedné a podmínka pro *TriggerType* FIRST je splněna.

Poznámka:

1. Z kroku "12" na stránce 323 (kde jsou zprávy spouštěče generovány jako výsledek některé události, která není doručena do fronty aplikací), se zpráva spouštěče nevloží jako součást pracovní jednotky. Pokud je *TriggerType* MQTT_EVERY a je-li ve frontě aplikace jedna nebo více zpráv, vygeneruje se pouze jedna zpráva spouštěče.
2. Pokud produkt WebSphere MQ segmentuje zprávu během operace MQPUT, událost spouštěče nebude zpracována, dokud nebudou všechny segmenty úspěšně umístěny do fronty. Nicméně jakmile jsou segmenty zprávy ve frontě, produkt WebSphere MQ je považuje za individuální zprávy pro účely spouštěče. Například jedna logická zpráva rozdělená na tři kusy způsobí zpracování pouze jedné události triggeru, když je první MQPUT a segmentovaná. Každý z těchto tří segmentů však způsobí, že vlastní události triggeru budou zpracovány, protože jsou přesunuty přes síť WebSphere MQ .

Řízení událostí spouštěče

Události spouštěče můžete řídit pomocí některých atributů, které definují frontu aplikací. Tyto informace také uvádí příklady použití typů spouštěčů: EVERY, FIRST a DEPTH.

Můžete povolit nebo zakázat spouštění a můžete vybrat počet nebo prioritu zpráv, které se počítají do události spouštěče. V části Atributy objektůje uveden úplný popis těchto atributů.

Příslušné atributy jsou:

TriggerControl
Tento atribut použijte k povolení a zakázání spouštění pro aplikační frontu.

TriggerMsgPriority
Minimální priorita, kterou musí zpráva mít, aby se napočítala ke spouštěcí události. Pokud je zpráva s prioritou nižší než *TriggerMsgPriority* doručena do fronty aplikací, správce front zprávu ignoruje, pokud určí, zda má být vytvořena zpráva spouštěče. Je-li parametr *TriggerMsgPriority* nastaven na hodnotu nula, budou všechny zprávy započítávají do události spouštěče.

TriggerType
Kromě spouštěče typu NONE (který zakazuje spouštění stejně jako nastavení *TriggerControl* na OFF), můžete použít následující typy spouštěčů k nastavení citlivosti fronty na události triggeru:

Každý	Událost spouštěče se vyskytne pokaždé, když přijde zpráva do aplikační fronty. Tento typ spouštěče použijte v případě, že chcete spustit více instancí aplikace.
PRVNÍ	Událost spouštěče se vyskytne pouze, když se počet zpráv v aplikační frontě změní z nuly na jeden. Tento typ spouštěče použijte, pokud chcete, aby se obsluhující program spustil při příchodu první zprávy do fronty, pokračovat až do doby, kdy nebudou další zprávy ke zpracování, pak ukončete. Frontu musíte vždy zpracovat, dokud nebude prázdná. Viz také "Speciální případ typu spouštěče FIRST" na stránce 326 .

DEPTH	Událost spouštěče se vyskytuje pouze tehdy, když se počet zpráv ve frontě aplikace dosáhne hodnoty atributu *TriggerDepth* . Typickým použitím tohoto typu spouštění je spuštění programu, když jsou přijaty všechny odpovědi na sadu požadavků. **Spouštění podle hloubky:** Se spuštěním hloubkou zakáže správce front spouštění (za použití atributu < xph> < pv>TriggerControl< /pv> < /xph>) poté, co vytvoří zprávu spouštěče. Vaše aplikace musí po provedení této události znovu povolit spuštění spouštěče (pomocí volání MQSET). Akce zakázání spouštěče není pod řízením synchronizačního bodu, takže spuštění nelze znovu povolit pomocí zálohování jednotky práce. Pokud program zálohuje požadavek na vložení, který způsobil událost spouštěče, nebo pokud se program ukončí, musíte znovu povolit spouštění pomocí volání MQSET nebo příkazu ALTER QLOCAL.

TriggerDepth
Počet zpráv ve frontě, které způsobí událost spouštěče při použití spouštěče podle hloubky.

Podmínky, které musí být splněny pro vytvoření zprávy spouštěče pro správce front, jsou popsány v tématu “Podmínky pro událost spouštěče” na stránce 321 .

Příklad použití typu spouštěče EVERY

Představte si aplikaci, která generuje požadavky na pojištění motorových vozidel. Aplikace může odeslat zprávy požadavku na celou řadu pojišťoven a pokaždé uvést stejnou odpověď do fronty. Může nastavit spouštěč typu KAŽDÉ v této odpovědi-na frontu tak, aby pokaždé, když přijde odpověď, mohla odpověď spustit instanci serveru pro zpracování odpovědi.

Příklad použití typu spouštěče FIRST

Zamysleme se nad organizací s mnoha pobočkami, které každý předává podrobnosti o pracovních dnech do hlavní kanceláře. Všichni to dělají ve stejnou dobu, na konci pracovního dne, a v kanceláři ústředí je aplikace, která zpracovává podrobnosti ze všech poboček. První zpráva, která má přijet do hlavní kanceláře, může způsobit událost spouštěče, která spustí tuto aplikaci. Tato aplikace by pokračovala ve zpracování, dokud ve frontě nejsou žádné další zprávy.

Příklad použití typu spouštěče DEPTH

Vezměme si cestovní kancelář aplikace, která vytváří jeden požadavek na potvrzení letenky rezervaci, potvrdit rezervaci pro hotelový pokoj, pronájem auta, a objednání některých cestujících kontroly. Aplikace může tyto položky rozdělit do čtyř zpráv vzniklých při zpracování požadavků, přičemž každá z nich bude posílat do samostatného místa určení. Může nastavit spouštěč typu DEPTH ve své odpovědi na frontu (s hloubkou nastaveným na hodnotu 4), takže bude restartována pouze v případě, že dorazily všechny čtyři odpovědi.

Pokud další zpráva (pravděpodobně z jiného požadavku) dorazí do fronty pro odpovědi před posledním ze čtyř odpovědí, je žádající aplikace spuštěna předčasně. Chcete-li se vyhnout tomu, že při použití funkce DEPTH ke shromažďování více odpovědí na požadavek, vždy pro každý požadavek použijte novou frontu pro odpověď.

Speciální případ typu spouštěče FIRST

S typem spouštěče FIRST, pokud je zpráva ve frontě zpráv již doručena, když přijde jiná zpráva, správce front obvykle nevytvoří další zprávu spouštěče.

Avšak aplikace obsluhující frontu nemusí ve skutečnosti otevřít frontu (například, aplikace by mohla skončit, možná kvůli problému se systémem). Pokud bylo do objektu definice procesu zadáno nesprávné jméno aplikace, aplikace obsluhující frontu nevyzvedne žádnou ze zpráv. Pokud v těchto situacích přijde

do fronty aplikace jiná zpráva, není spuštěn žádný server ke zpracování této zprávy (a žádné další zprávy ve frontě).

K vyřešení tohoto stavu vytvoří správce front další zprávy spouštěče za následujících okolností:

- Pokud do fronty aplikací dorazí další zpráva, ale pouze v případě, že od správce front je vytvořena poslední spouštěcí zpráva pro danou frontu, pouze v případě, že uplynul předdefinovaný časový interval. Tento časový interval je definován v atributu správce front *TriggerInterval*. Jeho výchozí hodnota je 999 999 999 milisekund.
- V produktu WebSphere MQ for z/OSjsou pravidelně procházeny fronty aplikací, které pojmenují otevřenou inicializační frontu. Pokud *TRIGINT* milisekund uplynuly od odeslání poslední zprávy spouštěče a fronta splňuje podmínky pro událost triggeru a CURDEPTH je větší než nula, vygeneruje se zpráva spouštěče. Tento proces se nazývá backstop spouštějící.

Při rozhodování o hodnotě intervalu spouštěče, který má být použit ve vaší aplikaci, zvažte následující body:

- Pokud jste nastavili *TriggerInterval* na nízkou hodnotu a neexistuje žádná aplikace obsluhující aplikační frontu, typ spouštěče FIRST se může chovat jako typ spouštěče KAŽDÉ. Závisí na rychlosti, jakou jsou zprávy vloženy do aplikační fronty, která naopak může záviset na jiné aktivitě systému. Je tomu tak proto, že je-li interval spouštěče velmi malý, generuje se při každém vložení zprávy do aplikační fronty jiná zpráva spouštěče i přesto, že typ spouštěče je FIRST, nikoli KAŽDÉ. (Typ spouštěče FIRST s intervalem spouštěče nula je ekvivalentní typu spouštěče ECVERY.)
- Pokud jste v produktu WebSphere MQ for z/OS nastavili *TRIGINT* na nízkou hodnotu a neexistuje žádná aplikace obsluhující typ spouštěče PRVNÍ, spustí se spouštěcí program backstop zprávu spouštěče pokaždé, když dojde k periodickému skenování front aplikací, které pojmenují otevřené fronty inicializace.
- Je-li jednotka práce zálohována (viz Spouštěcí zprávy a jednotky práce) a interval spouštěče byl nastaven na vysokou hodnotu (nebo výchozí hodnotu), vygeneruje se jedna zpráva spouštěče, když je jednotka práce zálohována. Avšak, pokud jste nastavili interval spouštěče na nízkou hodnotu nebo na nulu (způsobuje typ spouštěče FIRST, aby se choval jako typ spouštěče EVERY), lze generovat mnoho zpráv spouštěče. Je-li jednotka práce zálohována, všechny zprávy spouštěče jsou stále k dispozici. Počet vygenerovaných zpráv spouštěče závisí na intervalu spouštěče. Je-li interval triggeru nastaven na nulu, vygeneruje se maximální počet zpráv.

Návrh aplikace, která používá spuštěné fronty

Nyní jste viděli, jak nastavit a řídit spouštění vašich aplikací. Zde je několik tipů, které byste měli zvážit při návrhu vaší aplikace.

Zprávy spouštěče a jednotky práce

Zprávy triggeru vytvořené z důvodu událostí triggeru, které nejsou součástí jednotky práce, se umístí do inicializační fronty, mimo jakoukoli jednotku práce, bez závislosti na jakýchkoli jiných zprávách a jsou k dispozici pro načtení okamžitě monitorem spouštěče.

Zprávy spouštěče vytvořené z důvodu událostí triggeru, které jsou součástí jednotky práce, jsou zpřístupněny na inicializační frontě při vyřešení transakce UOW, ať už je jednotka práce potvrzená nebo zálohovaná

Pokud správce front nevloží do inicializační fronty zprávu spouštěče, bude umístěna do fronty nedoručených zpráv (undelivered-message).

Poznámka:

1. Správce front počítá potvrzené i nepotvrzené zprávy, když vyhodnocuje, zda existují podmínky pro událost spouštěče.

   S spuštěním typu FIRST nebo DEPTH jsou zprávy spouštěče zpřístupněny, i když je jednotka práce zálohována tak, že je zpráva spouštěče vždy dostupná, když jsou splněny požadované podmínky. Předpokládejme například, že požadavek na vložení do jednotky práce pro frontu, která je spuštěna

se spouštěčem typu FIRST. To způsobí, že správce front vytvoří zprávu spouštěče. Pokud se vyskytne jiný požadavek na vložení z jiné pracovní jednotky, nezpůsobí to jinou událost spouštěče, protože se počet zpráv ve frontě aplikace nyní změnil z jednoho na dva, což nesplňuje podmínky pro událost triggeru. Nyní, když je zálohována první jednotka práce, ale druhá je potvrzena, je stále vytvořena zpráva spouštěče.

To znamená, že zprávy spouštěče se někdy vytvářejí, když podmínky pro událost triggeru *nejsou* splněny. Aplikace, které používají spouštěče, musí být vždy připraveny ke zpracování této situace. Doporučuje se použít volbu čekání s voláním MQGET a nastavit hodnotu *WaitInterval* na vhodnou hodnotu.

Vytvořené zprávy spouštěče jsou vždy k dispozici, ať už je jednotka práce zálohována nebo potvrzena.

2. Pro lokální sdílené fronty (tj. sdílené fronty ve skupině sdílení front) správce front počítá pouze potvrzené zprávy.

Získávání zpráv ze spuštěné fronty

Když navrhujete aplikace, které používají spouštěcí impuls, uvědomte si, že může existovat prodleva mezi monitorem spouštěčů spouštějícím program a dalšími zprávami, které budou k dispozici ve frontě aplikací. To se může stát, když se zpráva, která způsobí událost triggeru, potvrdí před ostatními.

Chcete-li povolit příchod zpráv, vždy použijte volbu wait, když používáte volání MQGET k odebrání zpráv z fronty, pro které jsou nastaveny podmínky spouštěče. Hodnota *WaitInterval* musí být dostatečná k tomu, aby mohla být nejdelší vhodná doba mezi vkládanou zprávou a potvrzenými potvrzenými volání. Pokud se zpráva dostaví ze vzdáleného správce front, bude tato doba ovlivněna:

- Počet zpráv, které byly vloženy před potvrzením
- Rychlost a dostupnost komunikačního spojení
- Velikosti zpráv

Příklad situace, kdy byste měli použít volání MQGET s volbou wait, považujte za stejný příklad, jaký jsme použili při popisování jednotek práce. Toto byl požadavek na vložení do jednotky práce pro frontu spuštěnou s typem spouštěče FIRST. Tato událost způsobí, že správce front vytvoří zprávu spouštěče. Pokud se vyskytne jiný požadavek na vložení z jiné pracovní jednotky, nezpůsobí to jinou událost triggeru, protože se počet zpráv ve frontě aplikací nezměnil od nuly na jeden. Nyní, když je zálohována první jednotka práce, ale druhá je potvrzena, je stále vytvořena zpráva spouštěče. Zpráva spouštěče se tedy vytvoří v době, kdy je zálohována první jednotka práce. Pokud dojde k závažnému zpoždění před potvrzením druhé zprávy, může pro ni být spuštěna spuštěná aplikace.

Při spuštění typu DEPTH může dojít k prodlevě i v případě, kdy jsou všechny relevantní zprávy nakonec potvrzeny. Předpokládejme, že atribut fronty *TriggerDepth* má hodnotu 2. Když dvě zprávy dorazí do fronty, druhá způsobí, že se vytvoří zpráva spouštěče. Je-li však druhá zpráva první zpráva, která má být potvrzena, je v té době k dispozici zpráva spouštěče. Monitor spouštěčů spustí program serveru, ale program může načíst pouze druhou zprávu, dokud nebude potvrzena první zpráva. Je tedy možné, že program bude muset čekat na zpřístupnění první zprávy.

Navrhněte aplikaci tak, aby byla ukončena, pokud nejsou k dispozici žádné zprávy pro načtení v době, kdy vyprší interval čekání. Pokud se jedna nebo více zpráv dostaví později, spoléhejte se na to, že se vaše aplikace bude zpracovávat. Tato metoda zabraňuje tomu, aby aplikace byly nečinné a zbytečně využívám prostředky.

Inicializační fronta zpracovává monitory spouštěčů

Pro správce front je monitor spouštěčů stejně jako jakákoli jiná aplikace, která obsluhuje frontu. Avšak, monitor spouštěčů obsluhuje inicializační fronty.

Monitor spouštěčů je obvykle spouštěný program. Když zpráva spouštěče dorazí do inicializační fronty, monitor spouštěčů tuto zprávu načte. Používá informace ve zprávě k vydání příkazu ke spuštění aplikace, která má zpracovat zprávy ve frontě aplikací.

Monitor spouštěčů musí předat dostatečné informace programu, že se spouští, aby mohl program provádět správné akce ve správné aplikační frontě.

Inicializátor kanálu je příkladem speciálního typu monitoru spouštěčů pro agenty kanálu zpráv. V této situaci však musíte použít buď typ spouštěče FIRST nebo DEPTH.

Monitory spouštěčů na systémech UNIX a Windows

Toto téma obsahuje informace o monitorech spouštěčů, které jsou k dispozici na systémech UNIX a Windows .

Pro prostředí serveru jsou k dispozici následující monitory spouštěčů:

amqstrg0
Toto je ukázkový monitor spouštěčů, který poskytuje část funkce poskytované příkazem **runmqtrm**. Další informace o parametru amqstrg0viz "Ukázkové programy pro distribuované platformy" na stránce 93 .

runmqtrm
Syntaxe tohoto příkazu je **runmqtrm** [*-m QMgrName*] [*-q InitQ*], kde QMgrName je správce front a InitQ je inicializační fronta. Výchozí fronta je SYSTEM.DEFAULT.INITIATION.QUEUE ve výchozím správci front. Vyvolá programy pro odpovídající zprávy spouštěče. Tento monitor spouštěčů podporuje výchozí typ aplikace.

Příkazový řetězec předaný monitorem spouštěčů do operačního systému je sestaven takto:

1. *ApplId* z příslušné definice PROCESS (je-li vytvořena)
2. Struktura MQTMC2 ohraničená dvojitými uvozovkami
3. *EnvData* z příslušné definice PROCESS (je-li vytvořena)

kde *ApplId* je jméno programu, který má být spuštěn, jak by byl zadán na příkazovém řádku.

Předaný parametr je struktura znaků MQTMC2 . Je vyvolán příkazový řetězec, který má tento řetězec přesně tak, jak je uveden ve dvojitých uvozovkách, aby příkaz systému akceptoval tento řetězec jako jeden parametr.

Monitor spouštěčů se nepodívá, zda se v inicializační frontě nachází jiná zpráva až do dokončení právě spuštěného aplikačního serveru. Pokud má aplikace mnoho práce, monitor spouštěčů nemusí být schopen držet krok s počtem příchozích zpráv, které přicházejí do systému. Máte dvě možnosti:

- Mají spuštěné více monitorů spouštěčů
- Spustit spuštěné aplikace na pozadí

Pokud máte spuštěnu více monitorů spouštěčů, můžete řídit maximální počet aplikací, které mohou být spuštěny v libovolném okamžiku. Spustíte-li aplikace na pozadí, nebude produkt WebSphere MQ ukládat žádné omezení v počtu aplikací, které lze spustit.

Chcete-li spustit spuštěnou aplikaci na pozadí v systémech Windows , zadejte do pole *ApplId* předponu názvu vaší aplikace pomocí příkazu START. Příklad:

```
START ?B AMQSECHA
```

Chcete-li spustit spuštěnou aplikaci na pozadí na systémech UNIX , vložte & na konec definice *EnvData* definice PROCESS.

Poznámka: Pokud má cesta v systému Windows mezery jako část názvu cesty, měla by být uzavřena v uvozovkách (") aby bylo zajištěno, že se s ním zachází jako s jedním argumentem. Příklad: "C:\Program Files\Application Directory\Application.exe".

Níže je uveden příklad řetězce APPLICID, kde název souboru obsahuje mezery jako část cesty:

```
START "" /B "C:\Program Files\Application Directory\Application.exe"
```

Syntaxe příkazu Windows START v příkladu obsahuje prázdný řetězec uzavřený ve dvojitých uvozovkách. Příkaz START určuje, že se první argument v uvozovkách bude považovat za titulek nového příkazu. Chcete-li se ujistit, že systém Windows neudělá chybu v cestě aplikace pro argument 'title', přidejte před název aplikace řetězec názvu uzavřený do uvozovek do dvojitých uvozovek.

Pro klienta WebSphere MQ jsou k dispozici následující monitory spouštěčů:

runmqtmc
To je stejné jako runmqtrm s výjimkou toho, že odkazuje na knihovny klienta WebSphere MQ MQI.

Pro CICS
Monitor spouštěčů amqltmc0 je poskytován pro systém CICS. Funguje stejným způsobem jako standardní monitor spouštěčů, runmqtrm, ale spustíte jej jiným způsobem a spouští transakce CICS .

Toto téma se vztahuje pouze na systémy Windows, UNIXa Linux .

Je dodáván jako program CICS , definujte jej se 4místným názvem transakce. Zadejte 4znakový název, abyste spustili monitor spouštěčů. Používá výchozího správce front (jak je uveden v souboru qm.ini , nebo v produktu WebSphere MQ for Windows, registru) a v SYSTEM.CICS.INITIATION.QUEUE.

Chcete-li použít jiného správce front nebo frontu, sestavte strukturu monitoru spouštěčů MQTMC2 : to vyžaduje, abyste zapsali program pomocí volání EXEC CICS START, protože struktura je příliš dlouhá, než aby bylo možné přidat jako parametr. Poté předejte strukturu MQTMC2 jako data do požadavku START pro monitor spouštěčů.

Použijete-li strukturu MQTMC2 , musíte do monitoru spouštěčů dodat pouze parametry *StrucId*, *Version*, *QName*a *QMgrName* , protože neodkazují na žádná jiná pole.

Zprávy se čtou z inicializační fronty a používají se ke spuštění transakcí CICS pomocí EXEC CICS START za předpokladu, že APPL_TYPE ve zprávě spouštěče je MQAT_CICS. Čtení zpráv z inicializační fronty se provádí pod řízením synchronizačního bodu CICS .

Zprávy jsou generovány, když se monitor spustí a zastaví, a když se vyskytne chyba. Tyto zprávy jsou odeslány do přechodné datové fronty CSMT.

Zde jsou dostupné verze monitoru spouštěčů:

Verze	**Použití**
amqltmc0	TXSeries pro systémy AIX, HP-UXa Sun Solaris verze 5.1
amqltmc4	TXSeries pro systém Windows, verze 5.1
amqltmcc	Verze vazby klienta monitoru spouštěčů CICS

Pokud potřebujete monitor spouštěčů pro jiná prostředí, napište program, který může zpracovávat zprávy spouštěče, které správce front vloží do inicializačních front. Takový program by měl provádět následující akce:

1. Chcete-li čekat na příchod zprávy do inicializační fronty, použijte volání MQGET.
2. Prověřte pole struktury MQTM zprávy spouštěče, abyste nalezli název aplikace, která se má spustit, a prostředí, ve kterém se spustí.
3. Zadejte spouštěcí příkaz specifický pro prostředí.
4. V případě potřeby převeďte strukturu MQTM na strukturu MQTMC2 .
5. Předejte strukturu MQTMC2 nebo MQTM do spuštěné aplikace. Tato akce může obsahovat uživatelská data.
6. Přidružte k frontě aplikací aplikaci, která má sloužit k obsluze této fronty. To provedete tak, že pojmenujete objekt definice procesu (je-li vytvořen) v atributu *ProcessName* ve frontě.

Další informace o rozhraní monitoru spouštěčů najdete v tématu MQTMC2.

Vlastnosti zpráv spouštěče

V následujících tématech jsou popsány některé další vlastnosti zpráv spouštěče.

Perzistence a priorita zpráv spouštěče

Zprávy spouštěče nejsou trvalé, protože pro ně není určen žádný požadavek.

Avšak podmínky pro generování spouštěcích událostí trvají, takže zprávy spouštěče jsou generovány, kdykoli jsou tyto podmínky splněny. Dojde-li ke ztrátě zprávy spouštěče, bude pokračovat existence zprávy aplikace ve frontě aplikací a zaručuje, že správce front vygeneruje zprávu spouštěče ihned po splnění všech podmínek.

Je-li jednotka práce odvolána, všechny zprávy spouštěče, které vygenerovala, se vždy doručí.

Zprávy spouštěče mají výchozí prioritu inicializační fronty.

Restart zpráv správce front a spouštěcí zprávy

Po restartu správce front, když je inicializační fronta otevřena pro vstup, může být do této inicializační fronty vložena zpráva spouštěče, pokud je fronta aplikací přidružená k této frontě zpráv a je definována pro spuštění.

Spouštět zprávy a změny atributů objektu

Spouštěcí zprávy se vytvářejí podle hodnot atributů spouštěče platných v době události triggeru.

Pokud není zpráva spouštěče k dispozici pro monitor spouštěčů do pozdější doby (protože zpráva, která způsobila, že byla generována, byla vložena do pracovní jednotky), žádné změny atributů triggeru mezitím nemají žádný vliv na zprávu spouštěče. Zejména zakázání spouštění nezabrání zpřístupnění zprávy spouštěče po jeho vytvoření. Také fronta aplikací nemusí již existovat v době zpřístupnění zprávy spouštěče.

Formát zpráv spouštěče

Formát zprávy spouštěče je definován strukturou MQTM.

To má následující pole, která správce front vyplní, když vytvoří zprávu spouštěče, pomocí informací v definicích objektů ve frontě aplikace a procesu přidruženého k této frontě:

StrucId
Identifikátor struktury.

Version
Verze struktury.

QName
Název fronty aplikací, na které došlo k události spouštěče. Když správce front vytvoří zprávu spouštěče, zaplní toto pole pomocí atributu *QName* ve frontě aplikací.

ProcessName
Název objektu definice procesu, který je přidružen k frontě aplikací. Když správce front vytvoří zprávu spouštěče, zaplní toto pole pomocí atributu *ProcessName* ve frontě aplikací.

TriggerData
Pole ve volném formátu pro použití monitorem spouštěčů. Když správce front vytvoří zprávu spouštěče, zaplní toto pole pomocí atributu *TriggerData* ve frontě aplikací. V libovolném produktu

WebSphere MQ s výjimkou produktu WebSphere 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* v objektu definice procesu identifikovaném v souboru *ProcessName*.

ApplId
Znakový řetězec, který identifikuje aplikaci, kterou má spustit monitor spouštěčů. Když správce front vytvoří zprávu spouštěče, vyplní toto pole pomocí atributu *ApplId* v objektu definice procesu identifikovaném v souboru *ProcessName*. Při použití monitoru spouštěčů CKTI nebo CSQQTRMN dodávaného produktem WebSphere MQ for z/OSje atribut *ApplId* objektu definice procesu CICS nebo IMS Identifikátor transakce.

EnvData
Znakové pole obsahující data související s prostředním prostředím pro použití monitorem spouštěčů. Když správce front vytvoří zprávu spouštěče, vyplní toto pole pomocí atributu *EnvData* v objektu definice procesu identifikovaném v souboru *ProcessName*. WebSphere MQ for z/OS-dodané monitory spouštěčů (CKTI nebo CSQQTRMN) toto pole nepoužívají, ale jiné monitory spouštěčů se jej mohou rozhodnout používat.

UserData
Znakové pole obsahující uživatelská data pro použití monitorem spouštěčů. Když správce front vytvoří zprávu spouštěče, vyplní toto pole pomocí atributu *UserData* v objektu definice procesu identifikovaném v souboru *ProcessName*. Toto pole lze použít k určení názvu kanálu, který má být spuštěn.

V produktu MQTMje uveden úplný popis struktury zprávy spouštěče.

Když spouštění nefunguje

Program se nespustí, pokud monitor spouštěčů nemůže spustit program nebo správce front nemůže doručit zprávu spouštěče. Například ID aplikátoru v objektu procesu musí určovat, že program má být spuštěn na pozadí; jinak monitor spouštěčů nemůže spustit program.

Pokud je vytvořena zpráva spouštěče, ale nelze ji umístit do inicializační fronty (například, protože je plná fronta nebo je délka zprávy spouštěče větší než maximální délka zprávy uvedená pro inicializační frontu), spustí se zpráva spouštěče místo na frontu nedoručených zpráv (nedoručená zpráva).

Pokud operace vložení do fronty nedoručených zpráv nemůže být úspěšně dokončena, bude zpráva spouštěče zahozena a do konzoly z/OS nebo do systémového operátora se odešle varovná zpráva nebo je do protokolu chyb.

Vložení zprávy spouštěče do fronty nedoručených zpráv může generovat zprávu spouštěče pro danou frontu. Tato druhá zpráva spouštěče je vyřazena, pokud přidá zprávu do fronty nedoručených zpráv.

Pokud je program úspěšně spuštěn, ale ukončí se před tím, než přijme zprávu z fronty, použijte obslužný program pro trasování (například CICS AUXTRACE, pokud je program spuštěn pod CICS) k vyhledání příčiny selhání.

Práce s MQI a klastry

Existují speciální volby na voláních a návratových kódech, které se vztahují ke klastrování.

Použijte následující odkazy, chcete-li zjistit více o možnostech, které jsou k dispozici na voláních a návratových kódech pro použití s klastry:

- “MQOPEN a klastry” na stránce 333
- “MQPUT, MQPUT1 a klastry” na stránce 334
- “MQINQ a klastry” na stránce 334
- “MQSET a klastry” na stránce 335
- “Návratové kódy” na stránce 335

Související pojmy

“Přehled rozhraní fronty zpráv” na stránce 187
Zde jsou uvedeny informace o komponentách rozhraní MQI (Message Queue Interface).

“Připojování k správci front a odpojování od něj” na stránce 198
Chcete-li používat programovací služby produktu WebSphere MQ , musí mít program připojení ke správci front. V této části se dozvíte, jak se připojit k správci front a odpojit je od správce front.

“Otevírání a zavírání objektů” na stránce 206
Tyto informace poskytují náhled na otevření a zavření objektů produktu WebSphere MQ .

“Vložení zpráv do fronty” na stránce 216
V této části se dozvíte, jak vložit zprávy do fronty.

“Získávání zpráv z fronty” na stránce 231
Tyto informace použijte k získání informací o získávání zpráv z fronty.

“Inquaning about a nastavení atributů objektu” na stránce 307
Atributy jsou vlastnosti, které definují charakteristiky objektu produktu WebSphere MQ .

“Potvrzení a zálohování jednotek práce” na stránce 310
Tyto informace popisují, jak potvrdit a zazálohovat všechny zotavitelné operace get a put, které se vyskytly v jednotce práce.

“Spuštění aplikací produktu IBM WebSphere MQ pomocí spouštěčů” na stránce 316
Informace o spouštěčích a o tom, jak spustit aplikace IBM WebSphere MQ pomocí spouštěčů.

MQOPEN a klastry

Fronta, do které je při otevření fronty klastru vložena zpráva nebo z ní načtena, závisí na volání MQOPEN .

Výběr cílové fronty

Pokud nezadáte název správce front v deskriptoru objektu, produkt MQOD, správce front vybere správce front, do kterého má odeslat zprávu. Pokud v deskriptoru objektu zadáte název správce front, budou zprávy vždy odeslány do vybraného správce front.

Pokud správce front vybírá cílového správce front, závisí výběr na volbách vázání, MQOO_BIND_* a pokud existuje lokální fronta. Pokud se jedná o lokální instanci fronty, je vždy otevřená jako předvolba pro vzdálenou instanci, pokud není atribut CLWLUSEQ nastaven na hodnotu ANY. V opačném případě závisí výběr na volbách vazby. Musí být zadán buď MQOO_BIND_ON_OPEN nebo MQOO_BIND_ON_GROUP , když používáte skupiny zpráv s klastry, aby se zajistilo, že všechny zprávy ve skupině budou zpracovány ve stejném cíli.

Pokud správce front vybírá cílového správce front, provádí to způsobem round-robin s použitím algoritmu správy pracovní zátěže; viz Vyrovnávání zátěže.

MQOO_BIND_ON_OPEN
Volba MQOO_BIND_ON_OPEN na volání MQOPEN určuje, že má být správce cílové fronty opraven. Použijte volbu MQOO_BIND_ON_OPEN , pokud existuje více instancí stejné fronty v rámci klastru. Všechny zprávy vkládané do fronty s určením manipulátoru objektu vráceného z volání produktu MQOPEN jsou směrovány do stejného správce front.

- Pokud mají zprávy afinity, použijte volbu MQOO_BIND_ON_OPEN . Je-li například celá dávka zpráv zpracovávána ve stejném správci front, zadejte při otevření fronty produkt MQOO_BIND_ON_OPEN . Produkt IBM WebSphere MQ opravuje správce front a přenosovou cestu, která má být přijata všemi zprávami vkládané do této fronty.
- Je-li zadána volba MQOO_BIND_ON_OPEN , musí být fronta znovu otevřena pro novou instanci fronty, která má být vybrána.

MQOO_BIND_NOT_FIXED
Volba MQOO_BIND_NOT_FIXED ve volání MQOPEN určuje, že cílový správce front není pevný. Zprávy zapsané do fronty s uvedením ovladače objektu vrácené z volání MQOPEN jsou směrovány do správce

front v MQPUT čase na základě zpráv. Volbu MQOO_BIND_NOT_FIXED použijte, pokud nechcete, aby všechny vaše zprávy byly zapsány do stejného cíle.

- Nezadávejte MQOO_BIND_NOT_FIXED a MQMF_SEGMENTATION_ALLOWED zároveň. Pokud tak učiníte, segmenty vaší zprávy mohou být dodány různým správcům front, rozptýleným v celém klastru.

MQOO_BIND_ON_GROUP
Umožňuje aplikaci požadovat, aby byla skupina zpráv alokována do stejné cílové instance. Tato volba je platná pouze pro fronty a má vliv pouze na fronty klastru. Je-li tato volba zadána pro frontu, která není frontou klastru, je tato volba ignorována.

- Skupiny jsou směrovány pouze do jednoho místa určení, je-li MQPMO_LOGICAL_ORDER na příkazu MQPUT určeno. Je-li zadán MQOO_BIND_ON_GROUP, ale zpráva není součástí skupiny, použije se místo toho chování BIND_NOT_FIXED.

MQOO_BIND_AS_Q_DEF
Pokud nezadáte ani MQOO_BIND_ON_OPEN, MQOO_BIND_NOT_FIXED nebo MQOO_BIND_ON_GROUP, standardní volba je MQOO_BIND_AS_Q_DEF. Použití MQOO_BIND_AS_Q_DEF způsobí, že vazba, která se použije pro popisovač fronty, bude převzata z atributu fronty DefBind .

Relevance voleb obslužného programu MQOPEN

Volby MQOPEN MQOO_BROWSE, MQOO_INPUT_*nebo MQOO_SET vyžadují, aby byla lokální instance fronty klastru MQOPEN úspěšná.

Volby MQOPEN , MQOO_OUTPUT, MQOO_BIND_*nebo MQOO_INQUIRE nevyžadují lokální instanci fronty klastru, aby uspěly.

Vyřešený název správce front

Když je název správce front vyřešen v MQOPEN čase, je vyřešený název vrácen do aplikace. Pokud se aplikace pokusí použít tento název při dalším volání produktu MQOPEN , může zjistit, že nemá autorizaci pro přístup k názvu.

MQPUT, MQPUT1 a klastry

Je-li MQOO_BIND_NOT_FIXED zadán na MQOPEN , rutiny správy pracovní zátěže zvolí, které místo určení MQPUT nebo MQPUT1 vyberte.

Je-li MQOO_BIND_NOT_FIXED zadán ve volání příkazu MQOPEN , každé následující volání příkazu MQPUT vyvolá rutinu správy pracovní zátěže a určí správce front, kterému má být zpráva odeslána. Místo určení a cesta, které mají být vybrány, jsou vybrány na základě zpráv po zprávě. Místo určení a cesta se mohou po vložení zprávy změnit, pokud se změní podmínky v síti. Volání MQPUT1 vždy funguje, jako kdyby MQOO_BIND_NOT_FIXED byl v platnosti, tj. vždy vyvolává rutinu správy pracovní zátěže.

Když rutina správy pracovní zátěže vybrala správce front, dokončí operaci vložení lokální správce front. Zprávu lze umístit do jiných front:

1. Je-li cílem lokální instance fronty, zpráva se umístí do lokální fronty.
2. Je-li cílem správce front v klastru, bude zpráva vložena do přenosové fronty klastru.
3. Je-li cílem správce front mimo klastr, bude zpráva vložena do přenosové fronty se stejným názvem jako má cílový správce front.

Je-li v volání MQOPEN zadán parametr MQOO_BIND_ON_OPEN , volání MQPUT nespouští rutinu správy pracovní zátěže, protože cíl a trasa již byly vybrány.

MQINQ a klastry

Která fronta klastru je inquired upon závisí na volbách, které kombinujete s MQOO_INQUIRE.

Než se můžete dotázat na frontu, otevřete ji pomocí volání MQOPEN a zadejte MQOO_INQUIRE.

Chcete-li se dotázat na frontu klastru, použijte volání MQOPEN a zkombinujte ostatní volby s MQOO_INQUIRE. Atributy, které mohou být dotazovány, závisí na tom, zda existuje lokální instance fronty klastru a jak je fronta otevřena:

- Kombinace MQOO_BROWSE, MQOO_INPUT_*nebo MQOO_SET s MQOO_INQUIRE vyžaduje lokální instanci fronty klastru, aby byla otevřená úspěšná. V tomto případě se můžete dotázat na všechny atributy, které jsou platné pro lokální fronty.
- Kombinace MQOO_OUTPUT s MQOO_INQUIREa uvedení žádné z předchozích voleb, instance otevřené je buď:
  - Instance na lokálním správci front, pokud existuje. V tomto případě se můžete dotázat na všechny atributy, které jsou platné pro lokální fronty.
  - Instance na jiném místě v klastru, pokud neexistuje žádná lokální instance správce front. V tomto případě mohou být dotazovány pouze následující atributy. Atribut QType má hodnotu MQQT_CLUSTER v tomto případě.
    - DefBind
    - DefPersistence
    - DefPriority
    - InhibitPut
    - QDesc
    - QName
    - QTYPE

Chcete-li se dotázat na atribut DefBind ve frontě klastru, použijte volání MQINQ s voličem MQIA_DEF_BIND. Vrácená hodnota je buď MQBND_BIND_ON_OPEN , nebo MQBND_BIND_NOT_FIXED, nebo MQBND_BIND_ON_GROUP. Při použití skupin s klastry musí být zadán buď MQBND_BIND_ON_OPEN nebo MQBND_BIND_ON_GROUP .

Chcete-li se dotázat na atributy CLUSTER a CLUSNL na lokální instanci fronty, použijte volání MQINQ s voličem MQCA_CLUSTER_NAME nebo voličem MQCA_CLUSTER_NAMELIST.

Poznámka: Pokud otevřete frontu klastru bez určení fronty, ke které je operace MQOPEN svázána, může se následující volání produktu MQINQ dotazovat na různé instance této fronty klastru.

Související pojmy

“Volba MQOPEN pro frontu klastru” na stránce 212

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

The MQOPEN option MQOO_SET option requires there to be a local instance of a cluster queue for MQSET to succeed.

Volání MQSET nelze použít k nastavení atributů fronty jinde v klastru.

Můžete otevřít lokální alias nebo vzdálenou frontu definovanou s atributem klastru a použít volání MQSET . Atributy lokálního aliasu nebo vzdálené fronty můžete nastavit. Pokud je cílová fronta frontou klastru definovanou v jiném správci front, nezáleží na tom, zda je cílová fronta definována.

Návratové kódy

Návratové kódy specifické pro klastry

MQRC_CLUSTER_EXIT_ERROR (2266 X'8DA')

Volání MQOPEN, MQPUTnebo MQPUT1 se vydává za účelem otevření fronty klastru nebo vložení zprávy do fronty. Uživatelská procedura pracovní zátěže klastru, definovaná atributem ClusterWorkloadExit správce front, se neočekávaně nezdaří nebo neodpovídá.

Zpráva je zapsána do systémového protokolu v produktu WebSphere MQ for z/OS , která poskytuje více informací o této chybě.

Následné volání MQOPEN, MQPUTa MQPUT1 pro tento popisovač fronty se zpracují, jako by byl atribut ClusterWorkloadExit prázdný.

MQRC_CLUSTER_EXIT_LOAD_ERROR (2267 X'8DB')

V systému z/OSnelze uživatelskou proceduru pracovní zátěže klastru načíst.

Zpráva je zapsána do systémového protokolu a zpracování pokračuje, jako by atribut ClusterWorkloadExit byl prázdný.

Na jiných platformách než z/OSse pro připojení ke správci front vydá volání MQCONN nebo MQCONNX . Volání se nezdaří, protože nelze načíst uživatelskou proceduru pracovní zátěže klastru definovanou atributem správce front ClusterWorkloadExit správce front.

MQRC_CLUSTER_PUT_INHIBITED (2268 X'8DC')

Pro frontu klastru je vydáno volání MQOPEN s volbami MQOO_OUTPUT a MQOO_BIND_ON_OPEN v platnosti. Všechny instance fronty v klastru jsou momentálně blokovány tím, že má atribut InhibitPut nastaven na hodnotu MQQA_PUT_INHIBITED. Vzhledem k tomu, že nejsou k dispozici žádné instance fronty pro příjem zpráv, volání MQOPEN se nezdaří.

Tento kód příčiny se objevuje pouze v případě, že jsou splněny obě následující podmínky:

- Neexistuje žádná lokální instance fronty. Existuje-li lokální instance, volání MQOPEN bude úspěšné i v případě, že lokální instance bude blokována.
- Pro frontu neexistuje žádná uživatelská procedura pracovní zátěže klastru, nebo je zde uživatelská procedura pracovní zátěže klastru, ale nevybírá instanci fronty. (Pokud uživatelská procedura pracovní zátěže klastru zvolí instanci fronty, bude volání MQOPEN úspěšné, a to i v případě, že je tato instance zakázána.)

Je-li u volání MQOPEN zadána volba MQOO_BIND_NOT_FIXED , může být volání úspěšné i v případě, že všechny fronty v klastru budou mít blokováno vkládání. Následující volání příkazu MQPUT se však může nezdařit, pokud jsou všechny fronty stále blokovány v době volání.

MQRC_CLUSTER_RESOLUTION_ERROR (2189 X'88D')

1. Volání MQOPEN, MQPUTnebo MQPUT1 se vydává za účelem otevření fronty klastru nebo vložení zprávy do fronty. Definici fronty nelze správně rozpoznat, protože je vyžadována odezva od správce front úplného úložiště, ale žádná není k dispozici.
2. Volání MQOPEN, MQPUT, MQPUT1 nebo MQSUB je vydáno pro objekt tématu s určením PUBSCOPE(ALL) nebo SUBSCOPE(ALL). Definici tématu klastru nelze správně rozpoznat, protože je vyžadována odezva od správce front úplného úložiště, ale žádná není k dispozici.

MQRC_CLUSTER_RESOURCE_ERROR (2269 X'8DD')

Volání MQOPEN, MQPUTnebo MQPUT1 je vydáno pro frontu klastru. Vyskytuje se chyba při pokusu o použití prostředku požadovaného pro klastrování.

MQRC_NO_DESTINATIONS_AVAILABLE (2270 X'8DE')

Bylo zadáno volání MQPUT nebo MQPUT1 , které má vložit zprávu do fronty klastru. V době volání již v klastru nejsou žádné instance fronty. MQPUT selže a zpráva se neodešle.

Chyba se může vyskytnout, pokud je MQOO_BIND_NOT_FIXED zadán ve volání MQOPEN , který otevírá frontu, nebo MQPUT1 se používá k vložení zprávy.

MQRC_STOPPED_BY_CLUSTER_EXIT (2188 X'88C')

Volání MQOPEN, MQPUTnebo MQPUT1 se vydává za účelem otevření nebo vložení zprávy do fronty klastru. Uživatelská procedura pracovní zátěže klastru odmítne volání.

Tvorba klientských aplikací

Co potřebujete vědět, chcete-li zapisovat klientské aplikace do produktu WebSphere MQ.

Aplikace mohou být sestaveny a spouštěny v prostředí klienta WebSphere MQ . Aplikace musí být sestavena a propojena s použitým klientem WebSphere MQ MQI. Způsob, jakým jsou aplikace sestaveny a vzájemně propojeny, se liší podle použité platformy a programovacího jazyka. Informace o tom, jak vytvářet klientské aplikace, viz “Sestavování aplikací pro klienty WebSphere MQ MQI” na stránce 342.

Aplikaci WebSphere MQ lze spustit v plném prostředí produktu WebSphere MQ a v prostředí klienta WebSphere MQ MQI beze změny kódu za předpokladu, že jsou splněny určité podmínky. Další informace o spouštění aplikací v prostředí klienta WebSphere MQ naleznete v příručce “Spuštění aplikací v prostředí klienta IBM WebSphere MQ MQI” na stránce 344.

Pokud používáte rozhraní MQI (Message Queue Interface) k zápisu aplikací ke spouštění v prostředí klienta WebSphere MQ MQI, existují některé další ovládací prvky k ukládání během volání MQI, aby bylo zaručeno, že zpracování aplikace produktu WebSphere MQ nebude narušeno. Další informace o těchto ovládacích prvcích naleznete v tématu “Použití rozhraní MQI (Message Queue Interface) v klientské aplikaci” na stránce 338.

Informace o přípravě a spuštění jiných typů aplikací jako klientských aplikací naleznete v následujících tématech:

- “Příprava a spuštění aplikací CICS a Tuxedo” na stránce 355
- “Příprava a spuštění aplikací Microsoft Transaction Server” na stránce 38
- “Příprava a spuštění aplikací WebSphere MQ JMS” na stránce 358

Související pojmy

“Koncepty vývoje aplikací” na stránce 7
K zápisu aplikací IBM WebSphere MQ můžete použít volbu procedurálních nebo objektově orientovaných jazyků. Informace o konceptech produktu IBM WebSphere MQ , které jsou užitečné pro vývojáře aplikací, naleznete v odkazech v tomto tématu.

“Rozhodování o tom, jaký programovací jazyk použít” na stránce 75
Tyto informace použijte k vyhledání informací o programovacích jazycích a rámcích podporovaných produktem IBM WebSphere MQa o některých aspektech jejich použití.

“Návrh aplikací produktu IBM WebSphere MQ” na stránce 86
Když se rozhodnete, jak aplikace mohou využívat výhody platforem a prostředí, které máte k dispozici, musíte se rozhodnout, jak budete používat funkce nabízené produktem WebSphere MQ.

“Ukázka programů WebSphere MQ” na stránce 92
Tato kolekce témat slouží k získání informací o ukázkových programech WebSphere MQ na různých platformách.

“Psaní front aplikace” na stránce 187
Prostřednictvím těchto informací můžete získat informace o vytváření aplikací pro řazení do front, připojování a odpojování od správce front, publikování a odběru a otevírání a zavírání objektů.

“Použití webových služeb v produktu WebSphere MQ” na stránce 913
Můžete vyvíjet aplikace produktu IBM WebSphere MQ pro webové služby pomocí přenosu IBM WebSphere MQ pro protokol SOAP nebo mostu produktu IBM WebSphere MQ pro protokol HTTP.

“Zapisování aplikací typu publikování/odběr” na stránce 266
Začněte zapisovat do aplikací WebSphere MQ publish/subscribe.

“Sestavení aplikace IBM WebSphere MQ” na stránce 412
Tyto informace použijte k seznámení se s aplikací produktu IBM WebSphere MQ na různých platformách.

“Obsluha chyb programu” na stránce 529
Tyto informace vysvětlují chyby související s voláními MQI MQI buď při volání, nebo při doručení její zprávy do konečného cíle.

Použití rozhraní MQI (Message Queue Interface) v klientské aplikaci

Tato kolekce témat bere v úvahu rozdíly mezi zápisem aplikace WebSphere MQ pro spuštění v prostředí klienta WebSphere MQ MQI a ke spuštění v úplném prostředí správce front WebSphere MQ .

Při návrhu aplikace zvažte, jaké ovládací prvky je třeba uložit během volání MQI, aby bylo zaručeno, že zpracování aplikace produktu WebSphere MQ nebude narušeno.

Omezení velikosti zprávy v aplikaci klienta

Správce front má maximální délku zprávy, ale maximální velikost zprávy, kterou můžete přenášet z klientské aplikace, je omezena definicí kanálu.

Atribut maximální délky zprávy (MaxMsgLength) správce front je maximální délka zprávy, kterou může tento správce front zpracovat.

Na jiných platformách než z/OSmůžete zvýšit atribut maximální délky zprávy správce front. Podrobnosti jsou uvedeny v části ALTER QMGR.

Hodnotu proměnné MaxMsgpro správce front můžete zjistit pomocí volání MQINQ.

Pokud se změní atribut Délka MaxMsg, nekontroluje se, zda již neexistují fronty a dokonce zprávy s délkou větší než nová hodnota. Poté, co změníte tento atribut, restartujte aplikace a kanály, abyste se ujistili, že změna nabyla platnosti. Není tedy možné, aby byly generovány nové zprávy, které překročí délku MaxMsgDélka správce front nebo fronty (pokud není povolena segmentace správce front).

Maximální délka zprávy v definici kanálu omezuje velikost zprávy, kterou můžete přenášet po připojení klienta. Pokud se aplikace WebSphere MQ pokusí použít volání MQPUT nebo volání MQGET se zprávou větší než je tento, vrátí se do aplikace chybový kód. Parametr maximální velikosti zprávy definice kanálu neovlivňuje maximální velikost zprávy, kterou lze spotřebovat pomocí funkce MQCB v rámci připojení klienta.

Výběr identifikátoru kódové sady znaků klienta nebo serveru (CCSID)

Použijte lokální CCSID pro klienta. Správce front provádí nezbytný převod. Použijte proměnnou prostředí MQCCSID k přepsání CCSID. Pokud vaše aplikace provádí více PUTs, pole CCSID a kódování deskriptoru MQMD lze přepsat po dokončení prvního PUT.

Data předaná v rámci MQI z aplikace do stubu klienta musí být v lokálním CCSID kódovaném pro klienta WebSphere MQ MQI. Vyžaduje-li připojený správce front data, která mají být převedena, provede tento převod kód podpory klienta ve správci front.

Klient jazyka Java v V7však může provést převod, pokud to správce front není schopen provést. Viz “třídy připojení klienta WebSphere MQ pro připojení klienta Java” na stránce 647

Kód klienta předpokládá, že znaková data, která přecházejí z rozhraní MQI v klientovi, jsou v CCSID konfigurovaném pro danou pracovní stanici. Pokud tento CCSID není podporovaným CCSID nebo není vyžadovaným identifikátorem CCSID, může být přepsán pomocí proměnné prostředí MQCCSID pomocí jednoho z těchto příkazů:

- V systému Windows:

```
SET MQCCSID=850
```

- Na systémech UNIX:

```
export MQCCSID=850
```

Je-li tento parametr nastaven v profilu, předpokládá se, že všechna data MQI jsou v kódové stránce 850.

Poznámka: Předpokladem o kódové stránce 850 se nevztahují na data aplikace ve zprávě.

Pokud vaše aplikace provádí více operací PUTs, které zahrnují záhlaví WebSphere MQ po deskriptoru zpráv (MQMD), uvědomte si, že pole CCSID a kódování deskriptoru MQMD se přepíší po dokončení první operace PUT.

Po prvním PUT obsahují tato pole hodnotu, kterou používá připojený správce front k převodu záhlaví WebSphere MQ . Ujistěte se, že aplikace resetuje hodnoty na hodnoty, které vyžaduje.

Použití MQINQ v klientovi aplication

Některé hodnoty dotazované pomocí MQINQ jsou upraveny kódem klienta.

CCSID
je nastaven na hodnotu CCSID klienta, nikoli na straně správce front.

MaxMsgDélka
je zredukován, pokud je omezen definicí kanálu. To bude nižší z těchto hodnot:

- Hodnota definovaná v definici fronty, nebo
- Hodnota definovaná v definici kanálu

Další informace najdete v tématu MQINQ.

Použití koordinace bodu synchronizace v aplikaci klienta

Aplikace spuštěná na základním klientovi může vydávat MQCMIT a MQBACK, ale rozsah řízení synchronizačního bodu je omezen na prostředky MQI. Externí správce transakcí můžete použít s rozšířeným transakčním klientem.

V rámci produktu WebSphere MQje jedna z rolí správce front v aplikaci nastavena na ovládací prvek synchronizačního bodu. Je-li aplikace spuštěna na základním klientovi produktu WebSphere MQ , může zadat MQCMIT a MQBACK, ale rozsah řízení synchronizačního bodu je omezen na prostředky MQI. Příkaz MQBEGIN produktu WebSphere MQ není platný v základním prostředí klienta.

Aplikace spuštěné v úplném prostředí správce front na serveru mohou koordinovat více prostředků (například databází) prostřednictvím monitoru transakcí. Na serveru můžete použít nástroj Transaction Monitor dodávaný s produkty WebSphere MQ nebo jiný monitor transakcí, jako např. CICS. Monitor transakcí nemůžete používat s aplikací základního klienta.

Můžete použít externího správce transakcí s rozšířeným transakčním klientem WebSphere MQ . Viz Co je rozšířený transakční klient? pro podrobnosti.

Použití dopředného čtení v aplikaci klienta

Čtení napřed můžete použít na klientovi, abyste umožnili odeslání netrvalých zpráv klientovi, aniž by aplikace klienta musela požadovat zprávy.

Když klient vyžaduje zprávu ze serveru, odešle požadavek na server. Odešle samostatný požadavek pro každou zprávu, kterou spotřebuje. Chcete-li zvýšit výkon klientů, kteří spotřebovávají přechodné zprávy, tím, že se neodešlou tyto zprávy požadavku, může být klient nakonfigurován tak, aby používal dopředné čtení. Čtení napřed umožňuje odesílání zpráv klientovi, aniž by aplikace musela požadovat jejich zpracování.

Použití dopředného čtení může zlepšit výkon při spotřebovávání přechodných zpráv z klientské aplikace. Toto zlepšení výkonu je k dispozici pro aplikace MQI i JMS. Klientské aplikace používající příkaz MQGET nebo asynchronní spotřeba využívají zlepšení výkonu při spotřebovávání přechodných zpráv.

Při volání MQOPEN s MQOH_READAHEAD klient WebSphere MQ povolí čtení napřed, pokud jsou splněny určité podmínky. Tyto podmínky zahrnují:

- Jak klient, tak i vzdálený správce front musí používat produkt WebSphere MQ verze 7 nebo novější.
- Klientská aplikace musí být zkompilována a propojena s knihovnami klienta WebSphere MQ MQI s podporou podprocesů.
- Kanál klienta musí používat protokol TCP/IP.
- Kanál musí mít nenulové nastavení SharingConversations (SHARECNV) v definici kanálu klienta i serveru.

Je-li povoleno čtení napřed, jsou zprávy odesílány do vyrovnávací paměti na klientovi s názvem vyrovnávací paměť dopředného čtení. Klient má vyrovnávací paměť dopředného čtení pro každou frontu, kterou má otevřené s povolenou dopředným čtením. Zprávy ve vyrovnávací paměti dopředného čtení nejsou trvalé. Klient pravidelně aktualizuje server informacemi o množství dat, které spotřeboval.

Ne všechny návrhy aplikací klienta jsou vhodné pro použití čtení napřed, protože ne všechny volby jsou podporovány pro použití. Je-li povoleno čtení napřed, musí být některé volby konzistentní mezi voláními MQGET. Pokud klient změní svá kritéria výběru mezi voláními MQGET, zprávy ukládané do vyrovnávací paměti dopředného čtení zůstávají uvízlé v paměti dopředného čtení klienta dopředného čtení. Další informace naleznete v tématu “Zlepšení výkonu přechodných zpráv” na stránce 247

Konfigurace dopředného čtení je řízena třemi atributy, MaximumSize, PurgeTimea UpdatePercentage, které jsou určeny ve stanze MessageBuffer konfiguračního souboru klienta WebSphere MQ .

Použití asynchronního vložení v aplikaci klienta

Při použití asynchronního vložení může aplikace vložit zprávu do fronty, aniž by čekala na odezvu správce front. Tuto akci můžete použít ke zlepšení výkonu systému zpráv v některých situacích.

Za normálních okolností, když aplikace vloží zprávu nebo zprávy do fronty pomocí příkazu MQPUT nebo MQPUT1, musí aplikace čekat na to, aby správce front potvrdil, že zpracoval požadavek MQI. Můžete zvýšit výkon systému zpráv, a to zejména u aplikací, které používají vazby klienta, a aplikací, které nasadíte velký počet malých zpráv do fronty, tím, že vyberete místo toho, že chcete asynchronně vkládat zprávy. Když aplikace asynchronně odešle zprávu, správce front nevrátí úspěch nebo selhání každého volání, ale můžete namísto toho zkontrolovat chyby pravidelně.

Chcete-li asynchronně vložit zprávu do fronty, použijte volbu MQPMO_ASYNC_RESPONSE v poli *Options* struktury MQPMO.

Pokud zpráva není vhodná pro asynchronní vložení, je vložena do fronty synchronně.

Při požadavku na asynchronní odezvu vložení pro volání MQPUT nebo MQPUT1nemusí CompCode a příčina MQCC_OK a MQRC_NONE nezbytně znamenat, že zpráva byla úspěšně vložena do fronty. Ačkoli úspěšné nebo neúspěšné ukončení jednotlivých volání MQPUT nebo MQPUT1 nemusí být okamžitě vráceno, první chyba, která se vyskytla při asynchronním volání, může být později určena voláním MQSTAT.

Podrobnější informace o MQPMO_ASC_NC_RESPONSE najdete v tématu Volby MQPMO.

Ukázkový program Asynchronous Put demonstruje některé funkce, které jsou k dispozici. Podrobné informace o funkcích a návrhu programu a o jeho spuštění najdete v tématu “Ukázkový program Asynchronous Put” na stránce 110.

Použití sdílení konverzací v aplikaci klienta

V prostředí, ve kterém je povoleno sdílení konverzací, mohou konverzace sdílet instanci kanálu MQI.

Sdílení konverzací je řízeno dvěma poli s názvem SharingConversations, z nichž jedna je součástí struktury definice kanálu (MQCD) a jedna z nich je součástí struktury výstupního parametru kanálu (MQCXP). Pole SharingConversations na disku MQCD je celočíselná hodnota určující maximální počet konverzací, které mohou sdílet instanci kanálu přidruženou k kanálu. Pole SharingConversations v MQCXP je logická hodnota označující, zda je instance kanálu momentálně sdílená.

V prostředí, ve kterém není povoleno sdílení konverzací, nová připojení klienta s určujícím identickým rozhraním MQCD nebudou sdílet instanci kanálu.

Nové připojení klientské aplikace bude sdílet instanci kanálu, jsou-li splněny následující podmínky:

- Konce připojení klienta i připojení k serveru jsou konfigurovány pro sdílení konverzací a tyto hodnoty nejsou potlačeny ukončením kanálu.
- Hodnota MQCD pro připojení klienta (zadaná v rámci volání MQCONNX klienta nebo z tabulky CCDT) přesně odpovídá hodnotě MQCD připojení klienta zadané v rámci volání MQCONNX klienta nebo v tabulce CCDT, když byla poprvé zavedena existující instance kanálu. Všimněte si, že původní objekt

MQCD mohl být následně změněn ukončením nebo dohadování kanálu, ale že porovnání je provedeno proti hodnotě, která byla dodána na klientský systém před provedením těchto změn.

- Limit konverzací sdílení na straně serveru není překročen.

Pokud se nové připojení klientské aplikace shoduje s kritérii pro spuštění sdílení instance kanálu s jinými konverzacemi, je toto rozhodnutí provedeno dříve, než se k této konverzaci zavolají všechny uživatelské procedury. Uživatelské procedury v této konverzaci nemohou změnit skutečnost, že sdílí instanci kanálu s jinými konverzací. Nejsou-li k dispozici žádné existující instance kanálu odpovídající nové definici kanálu, bude připojena nová instance kanálu.

Vyjednávání kanálů probíhá pouze pro první konverzaci v instanci kanálu; vyjednané hodnoty pro instanci kanálu jsou v této fázi fixní a nelze je změnit při zahájení následných konverzací. Autentizace TLS/SSL se také vyskytuje pouze pro první konverzaci.

Je-li hodnota MQCD SharingConversations během inicializace zabezpečení ukončena, odešle nebo přijme ukončení pro první konverzaci na soketu buď na připojení klienta, nebo na konci spojení mezi serverem a připojením serveru, bude inicializována nová hodnota po všech těchto ukončovacích procedurách, aby bylo možné určit hodnotu konverzací sdílení pro instanci kanálu (nejnižší hodnota bude mít přednost).

Je-li vyjednaná hodnota pro sdílení konverzací nula, instance kanálu se nikdy nesdílí. Další ukončovací programy, které nastavují toto pole na nulu, se také spouštějí na své vlastní instanci kanálu.

Je-li vyjednaná hodnota pro sdílení konverzací větší než nula, pak je MQCXP SharingConversations nastaveno na hodnotu TRUE pro následná volání ukončení, což znamená, že jiné uživatelské programy na této instanci kanálu lze zadat souběžně s tímto voláním.

Při zápisu ukončovacího programu kanálu zvažte, zda bude spuštěn na instanci kanálu, která může zahrnovat sdílení konverzací. Pokud by instance kanálu mohla zahrnovat sdílení konverzací, uvažujte o vlivu na jiné instance kanálu uživatelské procedury pro změnu polí MQCD. Všechna pole MQCD mají společné hodnoty ve všech konverzacích sdílení. Pokud se po vytvoření instance kanálu pokusí změnit pole MQCD, mohou se setkat s problémy, protože jiné instance ukončovacích programů spuštěných na instanci kanálu by se mohly pokoušet o změnu stejných polí ve stejnou dobu. Pokud by tato situace mohla nastat s vašimi ukončovacími programy, musíte serializovat přístup ke kódu MQCD ve svém kódu ukončení.

Pokud pracujete s kanálem, který je definován pro sdílení konverzací, ale nechcete, aby se sdílení stalo na určité instanci kanálu, nastavte hodnotu MQCD SharingConversations na 1 nebo 0, když inicializujete kanál na první konverzaci na instanci kanálu. Viz SharingConversations , kde najdete vysvětlení hodnot SharingConversations.

Příklad

Sdílení konverzací je povoleno.

Používáte definici kanálu připojení klienta, která uvádí výstupní program.

Při prvním spuštění tohoto kanálu pozmění uživatelský program některé z parametrů MQCD, když je inicializován. Tyto akce jsou zpracovávány kanálem, takže definice, se kterou kanál běží, se nyní liší od té, která byla původně dodána. Parametr SharingConversations MQCXP je nastaven na hodnotu TRUE.

Při příštím připojení aplikace k tomuto kanálu je konverzace spuštěna na instanci kanálu, která byla spuštěna dříve, protože má stejnou původní definici kanálu. Instance kanálu, ke které se aplikace připojí podruhé, je stejná instance jako při prvním připojení k aplikaci. V důsledku toho používá definice, které byly změněny ukončovacím programem. Když je výstupní program inicializován pro druhou konverzaci, přestože může pozměnit pole MQCD, *nepostupuje* podle toho kanálu. Tyto charakteristiky se vztahují na všechny následující konverzace, které sdílejí instanci kanálu.

Použití MQCONNX

Volání MQCONNX můžete použít k určení struktury definice kanálu (MQCD) ve struktuře MQCNO.

To umožňuje volající aplikaci klienta určit definici kanálu připojení klienta za běhu. Další informace naleznete v tématu Použití struktury MQCNO při volání MQCONNX. Když použijete MQCONNX, volání vydané na serveru závisí na úrovni serveru a konfiguraci modulu listener.

Když používáte MQCONNX z klienta, jsou ignorovány následující volby:

- VAZBA MQCNO_STANDARD_BINDING
- VAZBA MQCNO_FASTPATH_BINDING

Struktura MQCD, kterou lze použít, závisí na počtu verzí produktu MQCD, které používáte. Informace o verzích produktu MQCD (MQCD_VERSION) naleznete v tématu Verze MQCD. Strukturu MQCD můžete použít například k předávání programů uživatelských procedur na server. Používáte-li produkt MQCD verze 3 nebo novější, můžete použít strukturu k předání pole uživatelských procedur na server. Tuto funkci můžete použít k provedení více než jedné operace na stejné zprávě, jako je šifrování a komprese, přidáním ukončení pro každou operaci, spíše než úpravou existující uživatelské procedury. Pokud ve struktuře MQCD nezadáte žádné pole, budou zkontrolována jednotlivá výstupní pole. Další informace o programech výstupních bodů kanálu naleznete v části “Kanály-uživatelské programy pro kanály systému zpráv” na stránce 380.

Sdílené popisovače připojení na MQCONNX

Mezi různými podprocesy ve stejném procesu můžete sdílet manipulátory s použitím sdílených manipulátorů připojení.

Při zadání sdílené obslužné rutiny připojení může být manipulátor připojení vrácený z volání MQCONNX předán v následných voláních MQI na libovolném podprocesu v procesu.

Poznámka: Pro připojení ke správci front serveru, který nepodporuje sdílené obslužné rutiny připojení, můžete použít manipulátor sdíleného připojení na klientu WebSphere MQ MQI.

Další informace viz “Použití MQCONNX” na stránce 341.

Sestavování aplikací pro klienty WebSphere MQ MQI

Aplikace lze sestavovat a pracovat v prostředí klienta WebSphere MQ MQI. Aplikace musí být sestavena a propojena s použitým klientem WebSphere MQ MQI. Způsob, jakým jsou aplikace sestaveny a vzájemně propojeny, se liší podle použité platformy a programovacího jazyka.

Má-li být aplikace spuštěna v prostředí klienta, můžete ji zapsat do jazyků zobrazených v následující tabulce:

Tabulka 47. Programovací jazyky podporované v prostředí klienta

Platforma klienta	C	JAZYK C++	COBOL	pTAL	RPG	Visual Basic
AIX	Ano	Ano	Ano			
HP Integrity NonStop Server	Ano		Ano	Ano		
HP-UX	Ano	Ano	Ano			
Linux	Ano	Ano	Ano			
Solaris	Ano	Ano	Ano			
Windows	Ano	Ano	Ano			Ano

Pokyny týkající se propojování nebo sestavení klientských aplikací v těchto jazycích najdete v souvisejících tématech.

Propojení aplikací jazyka C s kódem klienta WebSphere MQ MQI

Při zápisu aplikace WebSphere MQ , která má být spuštěna na klientovi WebSphere MQ MQI, je třeba ji propojit se správcem front.

Aplikaci můžete připojit ke správci front dvěma způsoby:

1. Přímo, v takovém případě musí být správce front na stejné pracovní stanici jako vaše aplikace
2. Do souboru knihovny klienta, který vám poskytuje přístup ke správcům front na stejném nebo na jiné pracovní stanici.

Produkt WebSphere MQ poskytuje soubor knihovny klienta pro každé prostředí:

AIX
Knihovnu libmqic.a pro aplikace bez podprocesů nebo knihovnu libmqic_r.a pro aplikace s podprocesy.

HP-UX
Knihovna libmqic.sl pro aplikace bez podprocesů, nebo knihovna libmqic_r.sl pro aplikace s podprocesy.

Linux
Knihovnu libmqic.so pro aplikace bez podprocesů, nebo knihovnu libmqic_r.so pro aplikace s podprocesy.

Solaris
libmqic.so.

Chcete-li používat programy na pracovní stanici, na které je instalován pouze klient WebSphere MQ MQI pro Solaris, musíte programy překompilovat, abyste je propojovali s knihovnou klienta:

```
$ /opt/SUNWspro/bin/cc -o <prog> <prog> c -mt -lmqic \
-lsocket -lc -lnsl -ldl
```

Parametry musí být zadány ve správném pořadí, jak je zobrazeno.

Windows
MQIC32.LIB.

Propojení aplikací C++ s kódem klienta WebSphere MQ MQI

Můžete psát aplikace, které se mají spustit na klientovi v C + +. Metody sestavení se liší v závislosti na prostředí.

Informace o tom, jak propojit vaše aplikace C + +, najdete v tématu Sestavování programů WebSphere MQ C++.

Podrobné informace o všech aspektech použití jazyků C + + naleznete v tématu Použití jazyka C++

Propojení aplikací COBOL s kódem klienta IBM WebSphere MQ MQI

Po zápisu aplikace v jazyce COBOL, kterou chcete spustit na klientu IBM WebSphere MQ MQI, je třeba ji propojit s příslušnou knihovnou.

Produkt IBM WebSphere MQ poskytuje soubor knihovny klienta pro každé prostředí:

AIX
Propojte aplikaci COBOL bez podprocesů s knihovnou libmqicb.a nebo s aplikací s podporou podprocesů v jazyce COBOL s parametrem libmqicb_r.a.

HP-UX
Propojte aplikaci COBOL bez podprocesů s knihovnou libmqicb.sl nebo s aplikací v jazyce COBOL s podporou podprocesů libmqicb_r.sl.

Linux
Propojte aplikaci COBOL bez podprocesů s knihovnou libmqicb.so nebo s aplikací v jazyce COBOL s podporou podprocesů libmqicb_r.so.

Solaris
Propojte aplikaci COBOL bez podprocesů s knihovnou libmqicb.so nebo s aplikací v jazyce COBOL s podporou podprocesů libmqicb_r.so.

Windows
Propojte kód aplikace s knihovnou MQICCBB pro 32bitový jazyk COBOL. Klient IBM WebSphere MQ MQI pro produkt Windows nepodporuje 16bitový jazyk COBOL.

Propojení aplikací Visual Basic s kódem klienta WebSphere MQ MQI

Aplikace v jazyku Visual Basic lze propojit s kódem klienta WebSphere MQ MQI v systému Windows.

Propojte si aplikaci Visual Basic s následujícími soubory začlenění:

CMQB.bas
MQI

CMQBB.bas
MQAI

CMQCFB.bas
příkazy PCF

CMQXB.bas
Kanály

Nastavte volbu `mqtype=2` pro klienta v kompilátoru Visual Basic, abyste zajistili správný automatický výběr knihovny DLL klienta:

MQIC32.dll
Windows 2000, Windows XP a Windows 2003

Spuštění aplikací v prostředí klienta IBM WebSphere MQ MQI

Aplikaci produktu IBM WebSphere MQ lze spustit v úplném prostředí produktu IBM WebSphere MQ i v prostředí klienta IBM WebSphere MQ MQI beze změny kódu za předpokladu, že jsou splněny určité podmínky.

Tyto podmínky jsou následující:

- Aplikace se nemusí připojovat k více než jednomu správci front souběžně.
- Název správce front není uvozeno hvězdičkou (*) ve volání `MQCONN` nebo `MQCONNX` .
- Aplikace nemusí používat žádnou z výjimek uvedených v tématu Jaké aplikace jsou spuštěny na klientovi IBM WebSphere MQ MQI?

Poznámka: Knihovny, které používáte při linkování, určují prostředí, ve kterém musí být aplikace spuštěna.

Při práci v prostředí klienta IBM WebSphere MQ MQI mějte na paměti následující skutečnosti:

- Každá aplikace spuštěná v prostředí klienta IBM WebSphere MQ MQI má svá vlastní připojení k serverům. Aplikace vytvoří jedno připojení k serveru pokaždé, když vyšle volání `MQCONN` nebo `MQCONNX` .
- Aplikace odesílá a přijímá zprávy synchronně. To znamená čekat mezi okamžikem, kdy je volání vydáno na klientovi, a návratem kódu dokončení a kódu příčiny v rámci sítě.
- Všechny konverze dat provádí server, ale viz také MQCCSID , kde jsou informace o přepisu nastavení CCSID počítače.

Připojení klientských aplikací IBM WebSphere MQ MQI ke správcům front

Aplikace běžící v prostředí klienta IBM WebSphere MQ MQI 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 WebSphere MQ vydá volání MQCONN nebo MQCONNX, klient identifikuje, jak má být připojení navázáno. Je-li volání MQCONNX vydáno aplikací na klientu IBM WebSphere MQ , knihovna klienta MQI vyhledá informace o kanálu klienta v následujícím pořadí:

1. Pomocí obsahu polí *ClientConnOffset* nebo *ClientConnPtr* struktury MQCNO (je-li k dispozici). Tato pole identifikují strukturu definice kanálu (MQCD), která má být použita jako definice kanálu připojení klienta. Podrobnosti o připojení lze přepsat pomocí uživatelské procedury před připojením. Další informace viz téma "Odkazování na definice připojení pomocí předání před připojením z úložiště" na stránce 407.
2. Je-li nastavena proměnná prostředí MQSERVER, bude použit kanál, který definuje.
3. Je-li definován soubor `mqclient.ini` a obsahuje parametry ServerConnectionParms, použije se kanál, který definuje. Další informace naleznete v tématu Konfigurace klienta pomocí konfiguračního souboru a stanza CHANNELS konfiguračního souboru klienta.
4. Jsou-li nastaveny proměnné prostředí MQCHLLIB a MQCHLTAB, použije se tabulka definic kanálů klienta, na kterou se má odkazovat.
5. Je-li definován soubor `mqclient.ini` a obsahuje atributy ChannelDefinitionDirectory a ChannelDefinitionFile, tyto atributy se používají k vyhledání tabulky definic kanálů klienta. Další informace naleznete v tématu Konfigurace klienta pomocí konfiguračního souboru a stanza CHANNELS konfiguračního souboru klienta.
6. Nakonec, nejsou-li proměnné prostředí nastaveny, klient vyhledá tabulku definic kanálů klienta s cestou a názvem, které jsou zavedeny ze souboru `DefaultPrefix` v souboru `mqs.ini` . Pokud vyhledávání v tabulce definic klienta selže, klient použije následující cesty:
   - Systémy UNIX and Linux : `/var/mqm/AMQCLCHL.TAB`
   - Windows: `C:\Program Files\IBM\Websphere MQ\amqclchl.tab`

První z voleb popsaných v předchozím seznamu (s použitím polí *ClientConnOffset* nebo *ClientConnPtr* MQCNO) je podporováno pouze voláním MQCONNX. Pokud aplikace používá MQCONN místo MQCONNX, budou informace o kanálu vyhledány ve zbývajících pěti bodech v pořadí zobrazeném v seznamu. Pokud se klientovi nepodaří najít informace o kanálu, volání MQCONN nebo MQCONNX selže.

Název kanálu (pro připojení klienta) se musí shodovat s názvem kanálu připojení serveru definovaným na serveru pro volání MQCONN nebo MQCONNX, aby bylo možné provést úspěch.

Pokud obdržíte návratový kód MQRC_Q_MGR_NOT_AVAILABLE s chybovou zprávou v souboru protokolu chyb `AMQ9517 -Poškozený soubor`, prohlédněte si téma Migrační a tabulky definic kanálů klienta (CCDT).

Související pojmy

Tabulka definic kanálů klienta

Související úlohy

Konfigurace připojení mezi serverem a klientem

Související odkazy

SERVER MQSERVER

MQCHLLIB

KARTA MQCHLTAB

Připojení klientských aplikací ke správcům front pomocí proměnných prostředí

Informace o kanálu klienta lze zadat do aplikace spuštěné v prostředí klienta pomocí proměnných prostředí MQSERVER, MQCHLLIB a MQCHLTAB.

Podrobné informace o těchto proměnných najdete v tématech MQSERVER, MQCHLLIB a MQCHLTAB .

Připojení klientských aplikací ke správcům front pomocí struktury MQCNO

Definici kanálu můžete zadat ve struktuře definice kanálu (MQCD), která je dodána pomocí struktury MQCNO volání MQCONNX.

Další informace viz téma Použití struktury MQCNO při volání MQCONNX .

Připojení klientských aplikací ke správcům front pomocí tabulky definic kanálů klienta

Pokud použijete příkaz MQSC DEFINE CHANNEL, zadané podrobnosti se umístí do tabulky definic kanálů klienta (ccdt). Obsah parametru *QMgrName* volání MQCONN nebo MQCONNX určuje, ke kterému správci front se klient připojuje.

K tomuto souboru přistupuje klient, aby určil kanál, který bude aplikace používat. Existuje-li více než jedna vhodná definice kanálu, je volba kanálu ovlivněna atributy kanálu kanálu klienta (CLNTWGHT) a kanálu afinity připojení (AFFINITY).

Role tabulky definic kanálů klienta

Tabulka definic kanálů klienta (CCDT) obsahuje definice kanálů připojení klienta. To je zvláště užitečné, pokud se vaše klientské aplikace mohou připojovat k řadě alternativních správců front.

Tabulka definic kanálů klienta se vytvoří, když definujete správce front.

Poznámka: Stejný soubor může být použit více než jedním klientem IBM WebSphere MQ . K různým verzím tohoto souboru se přistupuje pomocí proměnných prostředí MQCHLLIB a MQCHLTAB IBM WebSphere MQ . Informace o proměnných prostředí naleznete v tématu Použití proměnných prostředí produktu WebSphere MQ .

Skupiny správců front v tabulce CCDT

V tabulce definic kanálů klienta (CCDT) můžete definovat sadu připojení jako *skupinu správců front*. Aplikaci můžete připojit ke správci front, který je součástí skupiny správců front. To lze provést tak, že zafixujete název správce front na volání `MQCONN` nebo `MQCONNX` s hvězdičkou.

Můžete se rozhodnout definovat připojení k více než jednomu serverovém počítači, protože:

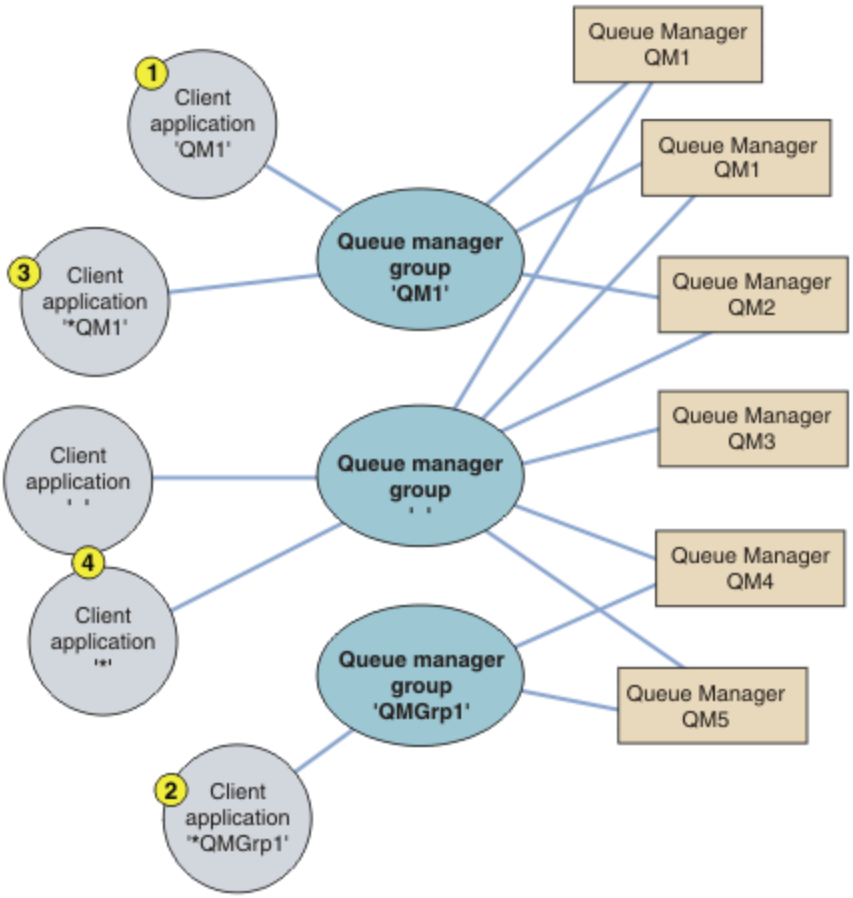
- Chcete-li zlepšit dostupnost, chcete připojit klienta k libovolné sadě správců front, kteří jsou spuštění.
- Chcete znovu připojit klienta ke stejnému správci front, k němuž došlo k úspěšnému připojení, ale připojit se k jinému správci front, pokud dojde k selhání připojení.
- Chcete-li znovu zkusit připojení klienta k jinému správci front, pokud dojde k selhání připojení, opětovným zadáním `MQCONN` v klientském programu chcete být znovu schopni připojení klienta k jinému správci front.
- Chcete automaticky znovu připojit připojení klienta k jinému správci front, pokud dojde k selhání připojení, aniž byste zapisoval nějaký kód klienta.
- Chcete-li automaticky znovu připojit připojení klienta k jiné instanci správce front s více instancemi, pokud dojde k selhání instance v pohotovostním režimu, bez zápisu jakéhokoliv kódu klienta.
- Chcete vyvážit připojení klienta přes několik správců front, s více klienty, kteří se připojují k některým správcům front, než u ostatních.
- Chcete rozložit opětovné připojení mnoha klientských připojení přes více správců front a v čase v případě, že vysoký objem připojení způsobí selhání.
- Chcete být schopni přesunout své správce front beze změny kódu aplikace klienta.
- Chcete zapsat aplikační programy klienta, které nepotřebují znát názvy správců front.

Není vždy vhodné připojit se k různým správcům front. Rozšířený transakční klient nebo klient Java na serveru WebSphere Application Servermůže například potřebovat připojení k předvídatelné instanci správce front. Třídy WebSphere MQ pro jazyk Java automatické opětovné připojování klientů nepodporují.

Skupina správců front je sada připojení definovaná v tabulce CCDT (Client Channel Definition table). Sada je definována svými členy se stejnou hodnotou atributu **QMNAME** ve svých definicích kanálů.

Obrázek 70 na stránce 347 je grafické znázornění tabulky připojení klienta zobrazující tři skupiny správců front, dvě pojmenované skupiny správců front zapsané v CCDT jako **QMNAME**(QM1) a **QMNAME**(QMGrp1)a jednu prázdnou nebo výchozí skupinu napsanou jako **QMNAME**(' ').

1. Skupina správce front QM1 má tři kanály připojení klienta, které se připojují ke správcům front QM1 a QM2. QM1 může být správce front s více instancemi umístěný na dvou různých serverech.
2. Výchozí skupina správců front má šest kanálů připojení klienta připojujících se ke všem správcům front.
3. QMGrp1 má kanály připojení klienta ke dvěma správcům front, QM4 a QM5.



Obrázek 70. Skupiny správců front

Čtyři příklady použití této tabulky připojení klienta jsou popsány v nápovědě k číslovaným klientským aplikacím v produktu Obrázek 70 na stránce 347.

1. V prvním příkladu předá aplikace klienta název správce front QM1jako parametr **QmgrName** do jeho volání MQCONN nebo MQCONNX MQI. Kód klienta produktu WebSphere MQ vybere odpovídající skupinu správců front QM1. Skupina obsahuje tři kanály připojení a klient WebSphere MQ MQI se pokusí o připojení k serveru QM1 pomocí každého z těchto kanálů až do nalezení modulu listener produktu WebSphere MQ pro připojení připojené ke spuštěnému správci front s názvem QM1.

   Pořadí pokusů o připojení závisí na hodnotě atributu AFFINITY připojení klienta a na váze kanálu klienta. V rámci těchto omezení je pořadí pokusů o připojení randomizováno, a to jak přes tři možné připojení, tak v průběhu času, aby se rozšířily načítání vytvářeného připojení.

   Volání MQCONN nebo MQCONNX vydané klientskou aplikací se zdaří, když je vytvořeno připojení k běžící instanci QM1.
2. Ve druhém příkladu aplikace klienta předává název správce front předponou s hvězdičkou, *QMGrp1 jako parametr **QmgrName** do jeho volání MQCONN nebo MQCONNX MQI. Klient WebSphere MQ vybere odpovídající skupinu správců front QMGrp1. Tato skupina obsahuje dva kanály připojení klienta a klient

WebSphere MQ MQI se pokusí o připojení ke správci front *any* s použitím kanálu postupně. V tomto příkladu musí klient WebSphere MQ MQI vytvořit úspěšné připojení; název správce front, k němuž se připojuje, je neaktuální.

Pravidlo pro pořadí pokusů o připojení je stejné jako u předchozích pokusů. Jediný rozdíl spočívá v tom, že při předběžném opravení názvu správce front s hvězdičkou klient indikuje, že název správce front není relevantní.

Volání MQCONN nebo MQCONNX vydané aplikací klienta je úspěšné, když je vytvořeno připojení ke spuštěné instanci jakéhokoli správce front připojeného k kanálům ve skupině správců front QMGrp1 .

3. Třetí příklad je v podstatě stejný jako druhý, protože parametr **QmgrName** je vložen hvězdičkou, *QM1. Následující příklad ilustruje, že nelze určit, ke kterému správci front se má připojení kanálu klienta připojovat, a to tak, že zkontrolujete atribut QMNAME v rámci jedné definice kanálu. Skutečnost, že atribut **QMNAME** definice kanálu je QM1, nepostačuje k tomu, aby bylo možné vytvořit připojení ke správci front s názvem QM1. Pokud vaše klientská aplikace přečíslí svůj parametr **QmgrName** s hvězdičkou, pak je kterýkoli správce front možným cílem připojení.

   V tomto případě volání MQCONN nebo MQCONNX vydaná aplikací klienta uspějí, když je ustanoveno spojení se spuštěnou instancí buď QM1 , nebo QM2.

4. Čtvrtý příklad ilustruje použití výchozí skupiny. V takovém případě aplikace klienta předá hvězdičku, '*' nebo prázdnou hodnotu '', jako argument **QmgrName** do jeho volání MQCONN nebo MQCONNX MQI. Podle konvence v definici kanálu klienta označuje prázdný atribut **QMNAME** výchozí skupinu správců front a parametr **QmgrName** prázdný nebo hvězdička se shoduje s prázdným atributem **QMNAME** .

   V tomto příkladu má výchozí skupina správců front připojení kanálů klienta ke všem správcům front. Vyberete-li výchozí skupinu správců front, může být aplikace připojena k libovolnému správci front ve skupině.

   Volání MQCONN nebo MQCONNX vydané aplikací klienta se zdaří, když je vytvořeno připojení ke spuštěné instanci libovolného správce front.

   Poznámka: Výchozí skupina se liší od výchozího správce front, ačkoli aplikace používá prázdný parametr **QmgrName** pro připojení buď k výchozí skupině správců front, nebo k výchozímu správci front. Koncept výchozí skupiny správců front je relevantní pouze pro aplikaci klienta a pro výchozí správce front v aplikaci serveru.

Definujte kanály připojení klienta pouze u jednoho správce front, včetně těch kanálů, které se připojují ke druhému nebo třetímu správci front. *Ne* je definuje ve dvou správcích front a poté se pokusí sloučit dvě definiční tabulky kanálu klienta. Klient může získat přístup pouze k jedné definiční tabulce kanálu klienta.

Příklady

Další informace naleznete v seznamu důvodů pro použití skupin správců front na začátku tématu. Jak pomocí skupiny správců front tyto možnosti poskytují?

Připojte se k některé z nich sady správců front.
Definujte skupinu správců front s připojeními ke všem správcům front v dané sadě a připojte se ke skupině pomocí parametru **QmgrName** s předponou hvězdičkou.

Znovu se připojte ke stejnému správci front, ale připojte se k jinému správci front, pokud je správce front připojen k poslední době nedostupný.
Definujte skupinu správců front jako dříve, ale nastavte atribut, **AFFINITY**(PREFERRED) na každé definici kanálu klienta.

Pokud připojení selže, zopakujte pokus o připojení k jinému správci front.
Připojte se ke skupině správců front a znovu zadejte volání MQCONN nebo MQCONNX MQI, je-li připojení přerušeno, nebo dojde k selhání správce front.

Automaticky se znovu připojit k jinému správci front, pokud připojení selže.
Připojte se ke skupině správců front pomocí volby MQCONNX **MQCNO** MQCNO_RECONNECT.

Automaticky se znovu připojte k jiné instanci správce front s více instancemi.
Postupujte stejně jako předchozí příklad. V tomto případě, chcete-li omezit skupinu správců front na připojení k instancím konkrétního správce front s více instancemi, definujte skupinu s připojeními pouze k instancím správce front s více instancemi.

Můžete také požádat klientskou aplikaci o vydání jeho volání MQCONN nebo MQCONNX MQI bez hvězdičky jako předpony s předponou **QmgrName** . Tímto způsobem se klientská aplikace může připojit pouze k uvedenému správci front. Nakonec můžete nastavit volbu **MQCNO** na hodnotu MQCNO_RECONNECT_Q_MGR. Tato volba přijímá nová připojení ke stejnému správci front, který byl dříve připojen. Tuto hodnotu můžete také použít k omezení opětovného připojení ke stejné instanci běžného správce front.

Vyvážit připojení klienta mezi správci front a dalšími klienty připojenými k některým správcům front než ostatním.
Definujte skupinu správců front a nastavte atribut **CLNTWGHT** na každé definici kanálu klienta, abyste nerovnoměrně distribuovali připojení.

Rozložit zátěž opakovaného připojení klienta nerovnoměrně a rozložit ji v čase po selhání připojení nebo správce front.
Postupujte stejně jako předchozí příklad. Klient WebSphere MQ MQI randomizuje přepojení mezi správci front a rozšiřuje opětovná připojení v průběhu času.

Přesuňte správce front beze změny kódu klienta.
CCDT izoluje aplikaci klienta od umístění správce front.

Můžete si vybrat buď rozdělení tabulky připojení klienta na každého klienta, nebo umístění tabulky CCDT na sdílený systém souborů pro každého klienta, na který se bude odkazovat. Případně můžete použít programovou verzi tabulky CCDT podporované v rámci volání MQI MQCONNX a volání služby pro předání tabulky CCDT do klientské aplikace.

Napište aplikaci typu klient, která nezná názvy správců front.
Použijte názvy skupin správců front a stanovte konvence pojmenování pro názvy skupin správců front, které jsou relevantní pro vaše klientské aplikace ve vaší organizaci, a odráží spíše architekturu vašich řešení než pojmenování správců front.

Připojování ke skupinám sdílení front
Tuto aplikaci můžete připojit ke správci front, který je součástí skupiny sdílení front. To lze provést použitím názvu skupiny sdílení front namísto názvu správce front v rámci volání MQCONN nebo MQCONNX.

Názvy skupin sdílení front jsou tvořeny nejvýše čtyřmi znaky. Název musí být v síti jedinečný a nesmí být shodný s žádným názvem správce front.

Definice kanálu klienta by měla používat generické rozhraní skupiny sdílení front pro připojení k dostupnému správci front ve skupině. Další informace naleznete v tématu Připojení klienta ke skupině sdílení front. Je provedena kontrola, aby se zajistilo, že se správce front, ke kterému se modul listener připojuje, bude členem skupiny sdílení front.

Příklady váhy kanálu a afinity
Tyto příklady ilustrují, jak jsou kanály připojení klienta vybrány, když se použije nenulová hodnota ClientChannelWeights .

Atributy kanálu ClientChannel a ConnectionAffinity řídí způsob, jakým jsou kanály připojení klienta vybrány, je-li k dispozici více než jeden vhodný kanál pro připojení. Tyto kanály jsou konfigurovány pro připojení k různým správcům front za účelem zajištění vyšší dostupnosti, vyrovnávání pracovní zátěže nebo obojího. Volání MQCONN, která by mohla vést k připojení k jednomu z několika správců front, musí před názvem správce front uvést hvězdičku podle popisu v:Příklady volání MQCONN: Příklad 1. Název správce front obsahuje hvězdičku (*).

Použitelné kandidátské kanály pro připojení jsou ta, kde atribut QMNAME odpovídá názvu správce front uvedenému v rámci volání MQCONN. Pokud mají všechny vhodné kanály pro připojení hodnotu ClientChannelVáha s hodnotou nula (výchozí), jsou tyto kanály vybrány v abecedním pořadí podle příkladu: Příklady volání MQCONN: Příklad 1. Název správce front obsahuje hvězdičku (*).

Následující příklady ilustrují, co se stane, když se použije nenulová hodnota `ClientChannelWeights` . Všimněte si, že vzhledem k tomu, že tato funkce zahrnuje pseudo-náhodný výběr kanálu, ukazují příklady posloupnost akcí, které se mohou stát, spíše než to, co rozhodně bude.

Příklad 1. Výběr kanálů, je-li parametr ConnectionAffinity nastaven na hodnotu PREFERRED
Tento příklad ukazuje, jak klient WebSphere MQ MQI vybere kanál z tabulky CCDT, kde je vlastnost ConnectionAffinity nastavena na hodnotu PREFERRED.

V tomto příkladu používá řada klientských počítačů tabulku CCDT (Client Channel Definition Table), kterou poskytuje správce front. Nástroje CCDT zahrnují kanály připojení klienta s následujícími atributy (zobrazené pomocí syntaxe příkazu DEFINE CHANNEL):

```
CHANNEL(A) QMNAME(DEV) CONNAME(devqm.it.company.example)
CHANNEL(B) QMNAME(CORE) CONNAME(core1.ops.company.example) CLNTWGHT(5) +
AFFINITY(PREFERRED)
CHANNEL(C) QMNAME(CORE) CONNAME(core2.ops.company.example) CLNTWGHT(3) +
AFFINITY(PREFERRED)
CHANNEL(D) QMNAME(CORE) CONNAME(core3.ops.company.example) CLNTWGHT(2) +
AFFINITY(PREFERRED)
```

Problémy aplikace MQCONN (*CORE)

Kanál A není kandidátem pro toto připojení, protože atribut QMNAME se neshoduje. Kanály B, C a D jsou identifikovány jako kandidáti a jsou umístěny v pořadí předvolby na základě jejich váhy. V tomto příkladu by pořadí mohlo být C, B, D. Klient se pokusí připojit ke správci front v adresáři core2.ops.company.example. Název správce front na této adrese se nekontroluje, protože volání MQCONN obsahovalo v názvu správce front hvězdičku.

Je důležité si uvědomit, že při každém připojení tohoto konkrétního klientského počítače k produktu `AFFINITY(PREFERRED)` dojde k umístění kanálů ve stejném počátečním pořadí předvoleb. To platí i v případě, že připojení pocházejí z různých procesů nebo v různou dobu.

V tomto příkladu není možné dosáhnout správce front v souboru core.2.ops.company.example . Klient se pokusí připojit k souboru core1.ops.company.example , protože kanál B je další v pořadí preferované pořadí. Kromě toho je kanál C degradován, aby se stal nejméně preferovaným.

Druhé volání MQCONN (*CORE) je vydáno stejnou aplikací. Kanál C byl degradován předchozím připojením, takže nejpreferovanější kanál je nyní B. Toto připojení je vytvořeno pro core1.ops.company.example.

Druhý počítač, který sdílí stejnou tabulku definic kanálů klienta, umístí kanály do jiného výchozího pořadí předvolby. Například D, B, C. Za normálních okolností se všemi pracujícími kanály jsou aplikace na tomto počítači připojeny k souboru core3.ops.company.example , zatímco jsou na prvním počítači připojeny k souboru core2.ops.company.example. To umožňuje vyrovnávání pracovní zátěže u velkého počtu klientů ve více správcích front a zároveň umožnit každému jednotlivému klientovi připojit se ke stejnému správci front, je-li k dispozici.

Příklad 2. Výběr kanálů, je-li parametr ConnectionAffinity nastaven na hodnotu NONE
Tento příklad ukazuje, jak klient WebSphere MQ MQI vybere kanál z tabulky CCDT, kde je parametr ConnectionAffinity nastaven na hodnotu NONE.

V tomto příkladu používá řada klientů tabulku CCDT (Client Channel Definition Table) poskytovanou správcem front. Nástroje CCDT zahrnují kanály připojení klienta s následujícími atributy (zobrazené pomocí syntaxe příkazu DEFINE CHANNEL):

```
CHANNEL(A) QMNAME(DEV) CONNAME(devqm.it.company.example)
CHANNEL(B) QMNAME(CORE) CONNAME(core1.ops.company.example) CLNTWGHT(5) +
AFFINITY(NONE)
CHANNEL(C) QMNAME(CORE) CONNAME(core2.ops.company.example) CLNTWGHT(3) +
AFFINITY(NONE)
CHANNEL(D) QMNAME(CORE) CONNAME(core3.ops.company.example) CLNTWGHT(2) +
AFFINITY(NONE)
```

Aplikace vydá MQCONN (*CORE). Stejně jako v předchozím příkladu se kanál A nebere v úvahu, protože se neshoduje s hodnotou QMNAME. Kanál B, C nebo D je vybrán na základě jejich váhy, s pravděpodobností

50%, 30% nebo 20%. V tomto příkladu může být zvolen kanál B. Neexistuje žádné trvalé pořadí preferované předvolby.

Je provedeno druhé volání MQCONN (*CORE). Opět platí, že je vybrán jeden ze tří použitelných kanálů se stejnou pravděpodobností. V tomto příkladu je zvolen kanál C. Nicméně core2.ops.company.example neodpovídá, takže je mezi zbývajícími kandidátskými kanály provedena jiná volba. Je vybrán kanál B a aplikace je připojena k souboru core1.ops.company.example.

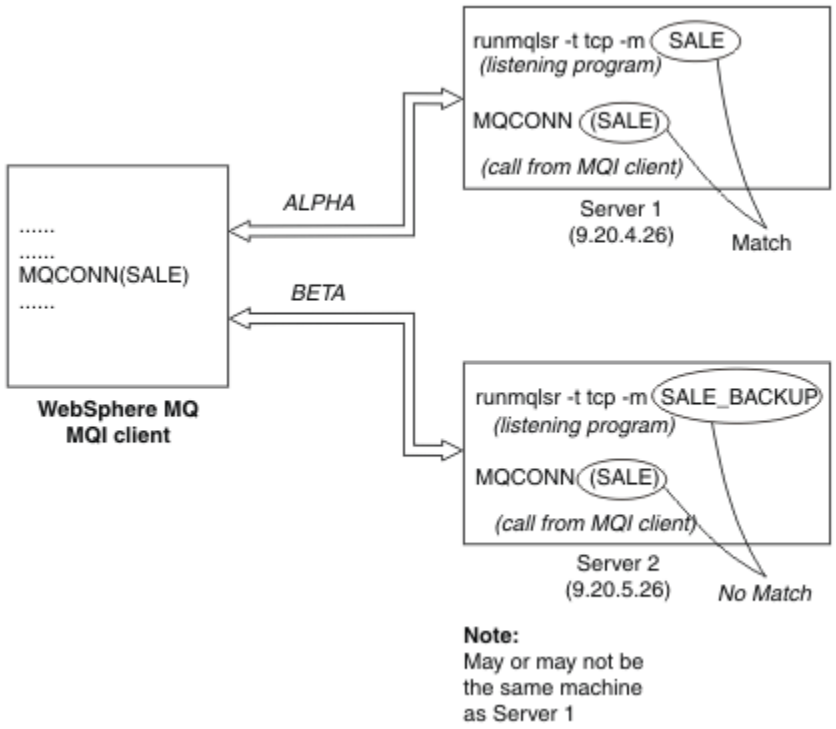
Při použití parametru AFFINITY (NONE) je každé volání MQCONN nezávislé na všech ostatních. Proto, když tato ukázková aplikace vytvoří třetí MQCONN (*CORE), může se pokusit o připojení prostřednictvím přerušeného kanálu C předtím, než jste vybrali jednu z hodnot B nebo D.

Příklady volání MQCONN

Příklady použití volání MQCONN pro připojení ke specifickému správci front nebo pro jednu ze skupin správců front.

V každém z následujících příkladů je síť stejná; existuje připojení definované na dvou serverech ze stejného klienta WebSphere MQ MQI. (V těchto příkladech může být místo volání MQCONN použit volání MQCONNX.)

Na serverovém počítači běží dva správci front, jeden s názvem SALE a druhý s názvem SALE_BACKUP.



Obrázek 71. Příklad MQCONN

Definice pro kanály v těchto příkladech jsou:

Definice SALE:

```
DEFINE CHANNEL(ALPHA) CHLTYPE(SVRCONN) TRPTYPE(TCP) +
DESCR('Server connection to WebSphere MQ MQI client')

DEFINE CHANNEL(ALPHA) CHLTYPE(CLNTCONN) TRPTYPE(TCP) +
CONNAME(9.20.4.26) DESCR('WebSphere MQ MQI client connection to server 1') +
QMNAME(SALE)

DEFINE CHANNEL(BETA) CHLTYPE(CLNTCONN) TRPTYPE(TCP) +
CONNAME(9.20.5.26) DESCR('WebSphere MQ MQI client connection to server 2') +
QMNAME(SALE)
```

Definice SALE_BACKUP:

```
DEFINE CHANNEL(BETA) CHLTYPE(SVRCONN) TRPTYPE(TCP) +
DESCR('Server connection to WebSphere 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

Jaké příklady příkazů MQCONN demonstrují
Příklady demonstrují použití více správců front jako záložního systému.

Předpokládejme, že komunikační propojení k serveru 1 je dočasně přerušeno. Je předvedeno použití více správců front jako záložního systému.

Každý příklad pokrývá různé volání MQCONN a poskytuje vysvětlení toho, co se děje ve specifickém ukázkovém příkladu za použití následujících pravidel:

1. Tabulka CCDT (Client Channel Definition CCDT) je skenována v abecedním pořadí názvů kanálů pro název správce front (pole QMNAME) odpovídající danému názvu správce front zpráv v rámci volání MQCONN.
2. Je-li nalezena shoda, použije se definice kanálu.
3. Došlo k pokusu o spuštění kanálu na počítači identifikovaném názvem připojení (CONNAME). Je-li tato operace úspěšná, bude aplikace pokračovat. Vyžaduje:
   - Modul listener, který má být spuštěn na serveru.
   - Listener, který má být připojen ke stejnému správci front jako ten, ke kterému se klient chce připojit (je-li zadán).
4. Pokud se pokus o spuštění kanálu nezdaří a v tabulce definic kanálů klienta je více než jedna položka (v tomto příkladu jsou dvě položky), prohledá se soubor kvůli další shodě. Je-li nalezena shoda, zpracování pokračuje v kroku 1.
5. Pokud není nalezena žádná shoda nebo pokud v tabulce definic kanálů klienta nejsou žádné další položky a kanál se nepodařilo spustit, aplikace se nedokáže připojit. Ve volání MQCONN je vrácen příslušný kód příčiny a kód dokončení. Aplikace může provést akci na základě toho, jak byly vráceny kódy příčiny a dokončení.

Příklad 1. Název správce front obsahuje hvězdičku (*)
V tomto příkladu se aplikace nevztahuje na správce front, k němuž se připojuje. Aplikace vydá volání MQCONN pro název správce front včetně hvězdičky. Je zvolen vhodný kanál.

Problémy aplikace:

```
MQCONN (*SALE)
```

V souladu s pravidly se v této instanci děje toto:

1. Tabulka definic kanálů klienta (CCDT) je skenována pro název správce front SALE, odpovídá volání MQCONN aplikace.
2. Jsou nalezeny definice kanálů pro ALPHA a BETA .
3. Má-li jeden kanál hodnotu CLNTWGHT 0, je tento kanál vybrán. Pokud má obě hodnotu CLNTWGHT hodnotu 0, je kanál ALPHA vybrán, protože je první v abecedním pořadí. Mají-li oba kanály nenulová hodnota CLNTWGHT, je jeden kanál náhodně vybrán na základě jeho váhy.
4. Je proveden pokus o spuštění kanálu.
5. Byl-li vybrán kanál BETA , pokus o spuštění je úspěšný.

6. Pokud byl vybrán kanál ALPHA , pokus o jeho spuštění NEBYL úspěšný, protože komunikační spoj je poškozen. Pak se použijí následující kroky:
   a. Jediným dalším kanálem pro název správce front SALE je BETA.
   b. Pokus o spuštění tohoto kanálu je úspěšný-tento pokus je úspěšný.
7. Kontrola, zda je modul listener spuštěný, ukazuje, že je spuštěný jeden. Není připojen ke správci front produktu SALE , ale protože má parametr volání MQI v něm obsažený znak hvězdička (*), nekontroluje se žádná kontrola. Aplikace je připojena ke správci front produktu SALE_BACKUP a pokračuje ve zpracování.

Příklad 2. Zadán název správce front
V tomto příkladu se aplikace musí připojit ke konkrétnímu správci front. Aplikace vydá volání MQCONN pro daný název správce front. Je zvolen vhodný kanál.

Aplikace vyžaduje připojení ke specifickému správci front s názvem SALE, jak je uvedeno v rámci volání MQI:

```
MQCONN (SALE)
```

V souladu s pravidly se v této instanci děje toto:

1. Tabulka definic kanálů klienta (CCDT) je skenována v posloupnosti názvů abecedního kanálu, pro název správce front SALE, odpovídá volání MQCONN aplikace.
2. První nalezená definice kanálu je ALPHA.
3. Pokus o spuštění kanálu byl proveden- *nebyl* úspěšný, protože komunikační spoj je poškozen.
4. Byla znovu naskenována tabulka definic kanálů klienta pro název správce front SALE a byl nalezen název kanálu BETA .
5. Pokus o spuštění kanálu byl proveden-došlo k úspěšnému pokusu o spuštění kanálu.
6. Kontrola, zda modul listener běží, ukazuje, že je zde jeden spuštěný, ale není připojen ke správci front produktu SALE .
7. V tabulce definic kanálů klienta nejsou žádné další položky. Aplikace nemůže pokračovat a přijímá návratový kód MQRC_Q_MGR_NOT_AVAILABLE.

Příklad 3. Název správce front je prázdný nebo hvězdička (*)
V tomto příkladu se aplikace nevztahuje na správce front, k němuž se připojuje. Aplikace vydá příkaz MQCONN s uvedením prázdného názvu správce front nebo hvězdičky. Je zvolen vhodný kanál.

To je zpracovkládáno stejným způsobem jako “Příklad 1. Název správce front obsahuje hvězdičku (*)” na stránce 352.

Poznámka: Pokud byla tato aplikace spuštěna v jiném prostředí než klient WebSphere MQ MQI a název byl prázdný, došlo k pokusu o připojení k výchozímu správci front. Toto *není* případ, kdy je spuštěn z prostředí klienta. Přístup ke správci front je ten, který je přidružen k modulu listener, ke kterému se kanál připojuje.

Problémy aplikace:

```
MQCONN ("")
```

, nebo

```
MQCONN (*)
```

V souladu s pravidly se v této instanci děje toto:

1. Tabulka definic kanálů klienta (CCDT) je skenována v posloupnosti názvů abecedního kanálu, pro název správce front, který je prázdný, odpovídá volání MQCONN aplikace.

2. Položka pro název kanálu ALPHA má název správce front v definici SALE. Tento parametr se *neshoduje* s parametrem volání MQCONN, který vyžaduje, aby byl název správce front prázdný.
3. Další položka je pro název kanálu BETA.
4. Definice `queue manager name` v definici je SALE. Opakuji, že toto *neodpovídá* parametru volání MQCONN, který vyžaduje, aby název správce front byl prázdný.
5. V tabulce definic kanálů klienta nejsou žádné další položky. Aplikace nemůže pokračovat a přijímá návratový kód MQRC_Q_MGR_NOT_AVAILABLE.

Spouštění v prostředí klienta

Zprávy odeslané aplikacemi produktu WebSphere MQ spuštěnými v klientech WebSphere MQ MQI přispívají ke spouštění stejným způsobem jako jiné zprávy a lze je použít ke spouštění programů na serveru i na straně klienta.

Spouštěcí impuls je podrobně vysvětlen v části Spouštěcí kanály.

Monitor spouštěčů a aplikace, které mají být spuštěny, musí být na stejném systému.

Výchozí charakteristiky spuštěné fronty jsou stejné jako výchozí charakteristiky v prostředí serveru. Zejména pokud v aplikaci klienta při vkládání zpráv do spuštěné fronty, která je lokální vzhledem ke správci front systému z/OS , nejsou zadány žádné volby řízení synchronizačního bodu MQPMO, zprávy jsou vloženy do jednotky práce. Je-li podmínka spouštěče splněna, zpráva spouštěče je vložena do inicializační fronty v rámci stejné jednotky práce a nemůže ji načíst monitor spouštěčů, dokud nebude ukončena jednotka práce. Proces, který se má spustit, se nespustí, dokud jednotka práce neskončí.

Definice procesu

Definici procesu musíte definovat na serveru, protože je to přidruženo k frontě, na které se spouští spouštěcí sada.

Objekt procesu definuje, co se má spustit. Pokud klient a server nejsou spuštění na stejné platformě, musí všechny procesy spuštěné monitorem spouštěčů definovat *ApplType*, jinak server převezme své výchozí definice (tj. typ aplikace, která je obvykle přidružena k počítači serveru) a způsobí selhání.

Je-li například monitor spouštěčů spuštěn na klientovi systému Windows a chce odeslat požadavek na server v jiném operačním systému, musí být MQAT_WINDOWS_NT definován jinak, jinak jiný operační systém používá své výchozí definice a proces selže.

monitor spouštěčů

Monitor spouštěčů poskytovaný produkty jiného typu nežz/OS WebSphere MQ se spouští v prostředí klienta pro systémy UNIX, Linux a Windows .

Chcete-li spustit monitor spouštěčů, zadejte jeden z těchto příkazů:

- Windows Linux UNIX Na platformách Windows, UNIX a Linux :

```
runmqtmc [-m QMgrName] [-q InitQ]
```

Výchozí inicializační fronta je SYSTEM.DEFAULT.INITIATION.QUEUE ve výchozím správci front. Inicializační fronta je tam, kde monitor spouštěčů vyhledává zprávy spouštěče. Pak volá programy pro odpovídající zprávy spouštěče. Tento monitor spouštěčů podporuje výchozí typ aplikace a je stejný jako `runmqtrm` , až na to, že spojuje knihovny klienta.

Příkazový řetězec sestavený monitorem spouštěčů je následující:

1. *ApplicId* z příslušné definice procesu. *ApplicId* je název programu, který má být spuštěn, tak jak by byl zadán na příkazovém řádku.
2. Struktura MQTMC2 je uzavřena v uvozovkách, která byla získána z inicializační fronty. Je spuštěn příkazový řetězec, který má tento řetězec přesně tak, jak je zadán, v uvozovkách, aby jej příkaz systému přijal jako jeden parametr.
3. *EnvrData* z příslušné definice procesu.

Monitor spouštěčů se nepodívá, zda se v inicializační frontě nachází jiná zpráva, dokud není dokončeno spuštění aplikace. Pokud má aplikace příliš mnoho práce, monitor spouštěčů nemusí držet krok s počtem příchozích zpráv, které přicházejí do styku. Existují dva způsoby, jak se s touto situací vypořádat:

1. Mají spuštěné více monitorů spouštěčů

   Rozhodnete-li se spustit více monitorů spouštěčů, můžete řídit maximální počet aplikací, které mohou být spuštěny v libovolném okamžiku.

2. Spustit spuštěné aplikace na pozadí

   Pokud zvolíte spuštění aplikací na pozadí, produkt WebSphere MQ nezavede žádné omezení počtu aplikací, které mohou být spuštěny.

Chcete-li spustit spuštěnou aplikaci na pozadí na systémech UNIX and Linux , musíte na konec *EnvrData* definice procesu vložit znak & (ampersand).

Aplikace CICS (jiné nežz/OS)

Aplikační program non-z/OS CICS , který vydává volání MQCONN nebo MQCONNX , musí být definován jako CEDA jako RESIDENT. Pokud znovu propojete aplikaci serveru CICS jako klienta, riskujete ztrátu podpory synchronizačních bodů.

Aplikační program non-z/OS CICS , který vydává volání MQCONN nebo MQCONNX , musí být definován jako CEDA jako RESIDENT. Chcete-li, aby byl rezidentní kód co nejmenší, můžete vytvořit odkaz na samostatný program, abyste mohli zavolat volání MQCONN nebo MQCONNX .

Je-li proměnná prostředí MQSERVER použita k definování připojení klienta, musí být uvedena v CICSENV.CMD .

Aplikace produktu WebSphere MQ lze spustit v prostředí serveru WebSphere MQ nebo v klientovi produktu WebSphere MQ bez změny kódu. V prostředí serveru WebSphere MQ však produkt CICS může vystupovat jako koordinátor synchronizačních bodů a vy budete používat příkaz EXEC CICS SYNCPOINT a EXEC CICS SYNCPOINT ROLLBACK, a nikoli MQCMIT a MQBACK. Pokud je aplikace CICS jednoduše přelněna jako klient, podpora synchronizačních bodů se ztratí. MQCMIT a MQBACK musí být použity pro aplikaci spuštěnou na klientovi WebSphere MQ MQI.

Příprava a spuštění aplikací CICS a Tuxedo

Chcete-li spustit aplikace CICS a Tuxedo jako klientské aplikace, použijte různé knihovny od těch, které používáte s aplikacemi serveru. ID uživatele, pod kterým je aplikace spuštěna, je také odlišné.

Chcete-li připravit aplikace CICS a Tuxedo, aby byly spouštěny jako klientské aplikace produktu WebSphere MQ MQI, postupujte podle pokynů v tématu Konfigurace rozšířeného klienta transakcí.

Všimněte si však, že informace, které se zabývají specificky pro přípravu aplikací CICS a Tuxedo, včetně ukázkových programů dodávaných s produktem WebSphere MQ, předpokládá, že připravujete aplikace ke spuštění na systému serveru WebSphere MQ . V důsledku toho se informace odkazují pouze na knihovny WebSphere MQ , které jsou určeny k použití na systému serveru. Při přípravě aplikací klienta je třeba provést následující akce:

- Použijte příslušnou systémovou knihovnu klienta pro vazby jazyků, které vaše aplikace používá. Například pro aplikace napsané v jazyce C v systému AIX, HP-UXnebo Solaris použijte knihovnu libmqic místo libmqm. Na systémech Windows použijte místo mqm.libknihovnu mqic.lib .
- Místo systémových knihoven serveru zobrazených v produktu Tabulka 48 na stránce 355, pro systémy AIX, HP-UXa Solaris a Tabulka 49 na stránce 356pro systémy Windows použijte ekvivalentní knihovny systému klienta. Není-li systémová knihovna serveru uvedena v těchto tabulkách, použijte stejnou knihovnu v klientském systému.

Tabulka 48. Systémové knihovny klienta v systému AIX, HP-UXa Solaris

Knihovna pro systém serveru WebSphere MQ	**Ekvivalentní knihovna k použití na klientském systému WebSphere MQ**
libmqmxa	libmqcxa

Tabulka 49. Systémové knihovny klienta v systémech Windows	
Knihovna pro systém serveru WebSphere MQ	**Ekvivalentní knihovna k použití na klientském systému WebSphere MQ**
mqmxa.lib	mqcxa.lib
mqmtux.lib	mqcxa.lib
mqmenc.lib	mqcxa.lib
mqmcics4.lib	mqccics4.lib

ID uživatele použité klientskou aplikací

Když spustíte aplikaci serveru WebSphere MQ pod CICS, normálně se přepne ze uživatele CICS na ID uživatele transakce. Pokud však spustíte aplikaci klienta WebSphere MQ MQI pod CICS, zachovají si privilegované oprávnění produktu CICS.

Ukázkové programy CICS a Tuxedo

Ukázkové programy CICS a Tuxedo pro použití v systémech AIX, HP-UX, Solaris a Windows.

Produkt Tabulka 50 na stránce 356 uvádí ukázkové programy CICS a Tuxedo, které jsou k dispozici pro použití v klientských systémech AIX, HP-UXa Solaris. Tabulka 51 na stránce 356 uvádí ekvivalentní informace pro klientské systémy Windows . Tabulky také zobrazují seznam souborů, které se používají pro přípravu a spuštění programů. Popis ukázkových programů viz “Ukázka transakce CICS” na stránce 113 a “Ukázky TUXEDO” na stránce 149.

Tabulka 50. Ukázkové programy pro klientské systémy AIX, HP-UXa Solaris		
Popis	**Zdroj**	**Spustitelný modul**
Program CICS	amqscic0.ccs	amqscicc
Soubor záhlaví pro program CICS	amqscih0.h	-
Program klienta Tuxedo pro vložení zpráv	amqstxpx.c	-
Klientský program Tuxedo pro získání zpráv	amqstxgx.c	-
Serverový program Tuxedo pro dva klientské programy	amqstxsx.c	-
Soubor UBBCONFIG pro programy Tuxedo	ubbstxcx.cfg	-
Soubor tabulky polí pro programy Tuxedo	amqstxvx.flds	-
Zobrazit soubor popisu pro programy Tuxedo	amqstxvx.v	-

Tabulka 51. Ukázkové programy pro klientské systémy Windows		
Popis	**Zdroj**	**Spustitelný modul**
Transakce CICS	amqscic0.ccs	amqscicc
Soubor záhlaví pro transakci CICS	amqscih0.h	-
Program klienta Tuxedo pro vložení zpráv	amqstxpx.c	-
Klientský program Tuxedo pro získání zpráv	amqstxgx.c	-
Serverový program Tuxedo pro dva klientské programy	amqstxsx.c	-
Soubor UBBCONFIG pro programy Tuxedo	ubbstxcx.cfg	-
Soubor tabulky polí pro programy Tuxedo	amqstxvx.fld	-

Tabulka 51. Ukázkové programy pro klientské systémy Windows (pokračování)		
Popis	**Zdroj**	**Spustitelný modul**
Zobrazit soubor popisu pro programy Tuxedo	amqstxvx.v	-
Makefile pro programy Tuxedo	amqstxmc.mak	-
Soubor ENVFILE pro programy Tuxedo	amqstxen.env	-

Chybová zpráva: AMQ5203, jak je upraveno pro aplikace CICS a Tuxedo

Když spustíte aplikace CICS nebo Tuxedo, které používají rozšířeného transakčního klienta, můžete vidět standardní diagnostické zprávy. Jeden z nich byl upraven pro použití s rozšířeným transakčním klientem

Zprávy, které se mohou zobrazit v souborech protokolu chyb produktu WebSphere MQ , jsou zdokumentovány v tématu Diagnostické zprávy: AMQ4000-9999. Zpráva AMQ5203 byla upravena pro použití s rozšířeným transakčním klientem. Zde je text upravené zprávy:

AMQ5203: Došlo k chybě při volání rozhraní XA.

Vysvětlení

Číslo chyby je & 2, kde hodnota 1 označuje, že zadaná hodnota příznaků '& 1 byla neplatná, 2 označuje, že došlo k pokusu o použití podprocesů s podporou podprocesů a bez podprocesů ve stejném procesu, 3 označuje, že došlo k chybě s dodaným názvem správce front' & 3 ', 4 označuje, že ID správce prostředků & 1 bylo neplatné, 5 označuje, že byl proveden pokus o použití druhého správce front s názvem'& 3 'když byl jiný správce front již připojen, 6 označuje, že správce transakcí byl volán, když aplikace není připojena ke správci front, 7 označuje, že volání XA bylo provedeno během dalšího volání, 8 označuje, že řetězec xa_info' & 4 'v volání xa_open obsahoval neplatnou hodnotu parametru pro název parametru' & 5 ', a 9 označuje, že řetězec xa_info' & 4 'v volání xa_open postrádá požadovaný parametr, název parametru' & 5 '.

Odezva uživatele

Opravte chybu a zkuste operaci znovu.

Příprava a spuštění aplikací Microsoft Transaction Server

Chcete-li připravit aplikaci MTS ke spuštění jako aplikaci klienta WebSphere MQ MQI, postupujte podle těchto pokynů, které odpovídají vašemu prostředí.

Obecné informace o vývoji aplikací serveru Microsoft Transaction Server (MTS), které přistupují k prostředkům produktu WebSphere MQ , naleznete v části týkající se MTS v centru nápovědy WebSphere MQ .

Chcete-li připravit aplikaci MTS ke spuštění jako aplikaci klienta WebSphere MQI MQ , proveďte jednu z následujících možností pro každou komponentu aplikace:

- Pokud komponenta používá vazby jazyka C pro rozhraní MQI, postupujte podle pokynů v části “Příprava programů jazyka C v systému Windows” na stránce 443 , ale propojte ji s knihovnou mqicxa.lib místo mqic.lib.
- Pokud komponenta používá třídy jazyka C++ produktu WebSphere MQ , postupujte podle pokynů v části “Sestavování programů C++ v systému Windows” na stránce 631 , ale propojte ji s knihovnou imqx23vn.lib namísto imqc23vn.lib.
- Pokud komponenta používá vazby jazyka Visual Basic pro rozhraní MQI, postupujte podle pokynů v příručce “Příprava programů jazyka Visual Basic v systému Windows” na stránce 447 , ale při definování projektu Visual Basic zadejte do pole **Argumenty podmíněné kompilace** hodnotu `MqType=3` .
- Pokud komponenta používá třídy automatizace produktu WebSphere MQ pro ActiveX (MQAX), definujte proměnnou prostředí GMQ_MQ_LIB s hodnotou `mqic32xa.dll` .

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

Příprava a spuštění aplikací WebSphere MQ JMS

Aplikace WebSphere MQ JMS můžete spouštět v režimu klienta pomocí aplikačního serveru WebSphere Application Server jako správce transakcí. Mohou se zobrazit určité varovné zprávy.

Chcete-li připravit a spustit aplikace WebSphere MQ JMS v režimu klienta s produktem WebSphere Application Server jako správce transakcí, postupujte podle pokynů v tématu “Použití tříd produktu WebSphere MQ pro službu JMS” na stránce 692.

Při spuštění aplikace klienta JMS produktu WebSphere MQ se mohou zobrazit následující varovné zprávy:

MQJE080
Nedostatečné licenční jednotky-spusťte příkaz setmqcap

MQJE081
Soubor obsahující informace o licenční jednotce je ve špatném formátu-spusťte příkaz settqcap

MQJE082
Soubor obsahující informace o jednotce licence nebyl nalezen-spusťte příkaz settmqcap

Uživatelské procedury, uživatelské procedury rozhraní API a instalovatelné služby produktu WebSphere MQ

Zařízení správce front lze rozšířit pomocí uživatelských procedur, uživatelských procedur rozhraní API nebo instalovatelných služeb. Toto téma obsahuje odkazy na informace o používání a vývoji těchto programů.

Úvod do způsobu použití uživatelských procedur, uživatelských procedur rozhraní API a instalovatelných služeb pro rozšíření zařízení správce front najdete v tématu Rozšíření zařízení správce front.

Další informace o psaní a kompilaci uživatelských procedur a instalovatelných službách naleznete v tématu “Uživatelské procedury pro psaní a kompilaci a instalovatelné služby” na stránce 358.

Související pojmy
Programy pro ukončení kanálů pro kanály MQI
Související odkazy
Popis uživatelské procedury rozhraní
Referenční informace o rozhraní instalovatelných služeb

Uživatelské procedury pro psaní a kompilaci a instalovatelné služby

Můžete psát a kompilovat uživatelské procedury bez odkazů na žádné knihovny produktu IBM WebSphere MQ v systémech UNIX, Linux a Windows.

Informace o této úloze

Windows Linux UNIX Toto téma platí pouze pro systémy Windows, UNIX and Linux . Podrobnosti o zápisu uživatelských procedur a instalovatelných služeb pro jiné platformy naleznete v příslušných tématech specifických pro platformu.

Je-li produkt IBM WebSphere MQ nainstalován v jiném než výchozím umístění, musíte vytvořit a zkompilovat uživatelské procedury bez odkazování na žádné knihovny produktu IBM WebSphere MQ .

V systémech Windows, UNIX and Linux můžete psát a kompilovat, aniž byste propojovali některou z těchto knihoven produktu IBM WebSphere MQ :

- mqmzf
- MQM

- mqmvx
- mqmvxd
- mqiová
- mkvl

Existující uživatelské procedury, které jsou propojeny s těmito knihovnami, budou pokračovat v práci, takže je v systému UNIX and Linux nainstalován produkt IBM WebSphere MQ ve výchozím umístění.

Postup

1. Zahrňte hlavičkový soubor cmqec.h .

   Začlenění tohoto souboru záhlaví automaticky zahrnuje soubory záhlaví cmqc.h, cmqxc.h a cmqzc.h .
2. Zadejte uživatelskou proceduru tak, aby byla volání MQI a DCI prováděna prostřednictvím struktury MQIEP. Další informace o struktuře MQIEP naleznete v tématu Struktura MQIEP.
   - Instalovatelné služby
     - Pomocí parametru **Hconfig** lze odkazovat na volání MQZEP.
     - Před použitím parametru **Hconfig** je třeba zkontrolovat, zda první 4 bajty **Hconfig** odpovídají struktuře **StrucId** struktury MQIEP.
     - Další informace o zápisu instalovatelných komponent služeb najdete v tématu MQIEP.
   - Uživatelské procedury rozhraní API
     - Pomocí parametru **Hconfig** lze odkazovat na volání MQXEP.
     - Před použitím parametru **Hconfig** je třeba zkontrolovat, zda první 4 bajty **Hconfig** odpovídají struktuře **StrucId** struktury MQIEP.
     - Další informace o zápisu uživatelských procedur rozhraní API najdete v tématu “Zápis uživatelských procedur API” na stránce 373.
   - Uživatelské procedury kanálu
     - Pomocí argumentu **pEntryPoints** struktury MQCXP lze odkazovat na volání MQI a DCI.
     - Před použitím produktu **pEntryPoints**je třeba zkontrolovat, zda je číslo verze MQCXP verze 8 nebo vyšší.
     - Další informace o zápisu uživatelských procedur kanálů naleznete v části “Psaní programů výstupních bodů kanálu” na stránce 383.
   - Ukončení převodu dat
     - Použijte parametr **pEntryPoints** struktury MQDXP, aby ukazoval na volání MQI a DCI.
     - Před použitím produktu **pEntryPoints**je třeba zkontrolovat, zda je číslo verze MQDXP verze 2 nebo vyšší.
     - Můžete použít příkaz **crtmqcvx** a zdrojový soubor amqsvfc0.c k vytvoření kódu pro převod dat, který používá parametr **pEntryPoints** . Další informace jsou uvedeny v tématech “Zápis uživatelské procedury pro převod dat pro produkt WebSphere MQ for Windows” na stránce 404 a “Zápis uživatelské procedury pro převod dat pro produkt WebSphere MQ v systémech UNIX and Linux” na stránce 401.
     - Pokud máte existující uživatelské procedury pro převod dat, které byly vygenerovány pomocí příkazu **crtmqcvx** , je nutné ukončit uživatelskou proceduru pomocí aktualizovaného příkazu.
     - Další informace o zápisu uživatelských procedur pro převod dat naleznete v tématu “Zápis uživatelských procedur pro převod dat” na stránce 399.
   - Uživatelské procedury před připojením
     - Pomocí argumentu **pEntryPoints** struktury MQNXP lze odkazovat na volání MQI a DCI.
     - Před použitím produktu **pEntryPoints**je třeba zkontrolovat, zda je číslo verze MQNXP verze 2 nebo vyšší.

- Další informace o zápisu východů před připojením naleznete v tématu “Odkazování na definice připojení pomocí předání před připojením z úložiště” na stránce 407.

- Uživatelské procedury publikování
  - Pomocí argumentu **pEntryPoints** struktury MQPSXP lze odkazovat na volání MQI a DCI.
  - Před použitím produktu **pEntryPoints**je třeba zkontrolovat, zda je verze MQPXP verze ve verzi 2 nebo vyšší.
  - Další informace o zápisu uživatelských procedur publikování naleznete v tématu “Zápis a kompilace uživatelských procedur pro publikování” na stránce 409.
- Ukončení pracovní zátěže klastru
  - Pomocí argumentu **pEntryPoints** struktury MQWXP lze odkazovat na volání MQXCLWLN.
  - Před použitím produktu **pEntryPoints**je třeba zkontrolovat, zda je verze MQWXP verze 4 nebo vyšší.
  - Další informace o zápisu uživatelských procedur pracovní zátěže klastru najdete v tématu “Zápis a kompilace uživatelských procedur pracovní zátěže klastru” na stránce 410.

Například v uživatelské proceduře kanálu volání MQPUT postupujte takto:

```
pChannelExitParms -> pEntryPoints -> MQPUT_Call(pChannelExitParms -> Hconn,
                                                Hobj,
                                                &md,
                                                &pmo,
                                                messlen,
                                                buffer,
                                                &CompCode,
                                                &Reason);
```

Další příklady lze zobrazit v příručce “Ukázka programů WebSphere MQ” na stránce 92.

3. Zkompilujte uživatelskou proceduru:
   - Nepropojujte s knihovnami produktu IBM WebSphere MQ .
   - Nezahrnujte vestavěnou cestu RPath do žádných knihoven produktu IBM WebSphere 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 374.
     - Uživatelské procedury kanálu, uživatelské procedury publikování, ukončení pracovní zátěže klastru: “Kompilace uživatelských programů kanálů v systémech Windows, UNIX and Linux” na stránce 398.
     - Uživatelské procedury pro převod dat: “Zápis uživatelských procedur pro převod dat” na stránce 399.
4. Oďte uživatelskou proceduru v jednom z následujících míst:
   - Cesta k vašemu výběru při konfiguraci uživatelské procedury
   - Výchozí cesta k ukončení, ve specifickém instalačním adresáři. Například *MQ_DATA_PATH*/exits/installation2.
   - Výchozí cesta k ukončení

     Výchozí cesta k výstupní cestě je *MQ_DATA_PATH*/exits pro 32 bitových východů a *MQ_DATA_PATH*/exits64 pro 64bitové procedury. Tyto cesty můžete změnit v souboru qm.ini nebo mqclient.ini . Další informace najdete v tématu Cesta k uživatelské proceduře. V systémech Windows a Linuxmůžete pomocí Průzkumníka produktu WebSphere MQ změnit cestu:

     a. Klepněte pravým tlačítkem myši na název správce front.

     b. Klepněte na **Vlastnosti ...**

     c. Klepněte na **Uživatelské procedury**

     d. Do pole výchozí cesty východů zadejte cestu k adresáři, který obsahuje ukončovací program.

Je-li uživatelská procedura umístěna do specifického instalačního adresáře a do výchozího adresáře cesty, použije se k instalaci produktu WebSphere MQ uvedeného v cestě k dispozici specifický instalační adresář instalačního adresáře. Například, výstup je umístěn v `/exits/installation2` a v `/exits`, ale ne v `/exits/installation1`. Instalace produktu WebSphere MQ `installation2` používá ukončení z produktu `/exits/installation2`. Instalace produktu WebSphere MQ `installation1` používá ukončení z adresáře `/exits` .

5. Je-li to nezbytné, nakonfigurujte uživatelskou proceduru:
   - Instalovatelné služby: “Konfigurace služeb a komponent” na stránce 368.
   - Uživatelské procedury rozhraní API: “Konfigurace uživatelských procedur rozhraní API” na stránce 378.
   - Uživatelské procedury kanálu: “Konfigurace uživatelských procedur kanálu” na stránce 399.
   - Uživatelské procedury publikování: “Konfigurace uživatelských procedur publikování” na stránce 410.
   - Ukončení před připojením: “stanza PreConnect konfiguračního souboru klienta” na stránce 408.

Instalovatelné služby a komponenty pro systémy UNIX, Linux a Windows

Tento oddíl uvádí instalovatelné služby a funkce a komponenty, které jsou k nim přidruženy. Rozhraní pro tyto funkce je dokumentováno tak, že vy nebo dodavatelé softwaru můžete dodávat komponenty.

Hlavní důvody pro poskytování instalovatelných služeb produktu WebSphere MQ jsou:

- Chcete-li vám poskytnout flexibilitu při výběru toho, zda mají být použity komponenty poskytované produkty WebSphere MQ , nebo je můžete nahradit nebo je rozšiřovat s ostatními.
- Chcete-li dodavatelům umožnit účast, tím, že poskytnete komponenty, které mohou používat nové technologie, aniž byste provedli interní změny v produktech WebSphere MQ .
- Umožněte produkt WebSphere MQ využívat rychlejší a levnější využití nových technologií, a proto poskytovat produkty dříve a za nižší ceny.

Instalovatelné služby a *komponenty služeb* jsou součástí struktury produktu WebSphere MQ . Ve středu této struktury je ta část správce front, která implementuje danou funkci a pravidla přidružená k rozhraní MQI (Message Queue Interface). Tato centrální část vyžaduje řadu servisních funkcí nazývaných *instalovatelné služby*, aby mohla provést svou práci. Instalovatelné služby jsou:

- Autorizační služba
- služba názvů

Každá instalovatelná služba je související sadou funkcí implementovaných pomocí jedné nebo více *komponent služeb*. Každá komponenta je vyvolána pomocí veřejně dostupného rozhraní, které je k dispozici. To umožňuje nezávislým dodavatelům softwaru a dalším třetím stranám poskytovat instalovatelné komponenty k rozšíření nebo nahrazení těch, které jsou poskytovány produkty WebSphere MQ . Tabulka 52 na stránce 361 shrnuje služby a komponenty, které lze použít.

Tabulka 52. Souhrn komponent instalovatelné služby

instalovatelná služba	**Dodaná komponenta**	**Funkce**	**Požadavky**
Autorizační služba	správce oprávnění k objektu (OAM)	Poskytuje kontrolu autorizace pro příkazy a volání MQI. Uživatelé mohou napsat svou vlastní komponentu pro rozšíření nebo nahrazení OAM. Chcete-li například zkontrolovat, zda má ID uživatele oprávnění k otevření fronty.	(Předpokládá se vhodná oprávnění k autorizaci platformy)

Tabulka 52. Souhrn komponent instalovatelné služby (pokračování)			
instalovatelná služba	**Dodaná komponenta**	**Funkce**	**Požadavky**
služba názvů	Není	Poskytuje podporu pro správce front za účelem vyhledání názvu správce front, který vlastní určenou frontu. • Definované uživatelem **Poznámka:** Sdílené fronty musí mít nastaven atribut *Scope* nastaven na hodnotu CELL.	• třetí strana nebo uživatel zapsaný jménem uživatele

Rozhraní instalovatelných služeb je popsáno v tématu Referenční informace o rozhraní instalovatelných služeb.

Psaní komponenty služby

Tato sekce popisuje vztah mezi službami, komponentami, vstupními body a návraty k návratovému kódu.

Funkce a komponenty

Každá služba se skládá ze sady souvisejících funkcí. Např. služba názvů obsahuje funkci pro:

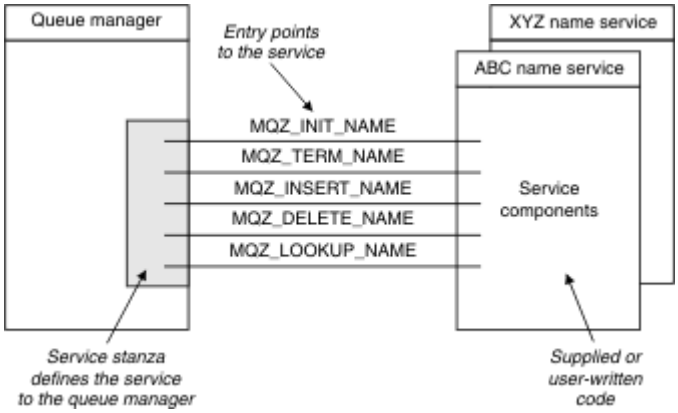
- Probíhá hledání názvu fronty a vrácení názvu správce front, ve kterém je fronta definována.
- Vložení názvu fronty do adresáře služby
- Odstranění názvu fronty z adresáře služby

Obsahuje také funkce inicializace a ukončení.

Instalovatelná služba je poskytována jednou nebo více komponentami služeb. Každá komponenta může provádět některé nebo všechny funkce, které jsou pro danou službu definovány. Například v produktu WebSphere MQ for AIXdodá komponenta autorizační služby, komponenta OAM, provede všechny dostupné funkce. Další informace viz “Rozhraní autorizační služby” na stránce 366. Komponenta je také odpovědná za správu veškerých podkladových prostředků nebo softwaru (například adresáře LDAP), které potřebuje k implementaci služby. Konfigurační soubory poskytují standardní způsob, jak načíst komponentu a určit adresy funkčních rutin, které poskytuje.

Obrázek 72 na stránce 363 ukazuje, jak jsou služby a komponenty související:

- Služba je definována pro správce front podle oddílů v konfiguračním souboru.
- Každá služba je podporována zadaným kódem ve správci front. Uživatelé nemohou tento kód změnit, a proto nemohou vytvořit své vlastní služby.
- Každá služba je implementována jednou nebo více komponentami. Tyto služby mohou být dodány spolu s produktem nebo uživatelem napsanými. Je možné vyvolat více komponent pro službu, přičemž každá z nich podporuje různá zařízení v rámci služby.
- Vstupní body spojují komponenty služeb s podpůrným kódem ve správci front.



Obrázek 72. Základní informace o službách, komponentách a vstupních bodech

Vstupní body

Každá komponenta služby je představována pomocí seznamu adres vstupního bodu rutin, které podporují konkrétní instalovatelnou službu. Instalovatelná služba definuje funkci, která má být prováděna každou rutinou.

Uspořádání komponent služeb, když jsou nakonfigurovány, definuje pořadí, ve kterém jsou vstupní body volány ve snaze vyhovět požadavku na službu.

V dodaném hlavičkovém souboru `cmqzc.h` mají dodávané vstupní body pro každou službu předponu MQZID_.

Jsou-li služby přítomny, služby se načtou v předdefinovaném pořadí. Následující seznam zobrazuje služby a pořadí, ve kterém jsou inicializovány.

1. `NameService`
2. `AuthorizationService`
3. `UserIdentifierService`

`AuthorizationService` je jediná služba, která je ve výchozím nastavení konfigurována. Pokud je chcete použít, konfigurujte produkty `NameService` a `UserIdentifierService` ručně.

Služby a komponenty služeb mají mapování jeden-na-jednoho nebo jeden-na-mnoho. Pro každou službu lze definovat více komponent služby. Na systémech UNIX and Linux se hodnota Service Service stanza `ServiceComponent` musí shodovat s hodnotou názvu servisní sekce v souboru qm.ini . V systému Windowsmusí hodnota klíče registru služeb produktu `ServiceComponent`odpovídat hodnotě klíče registru názvu a je definována jako: `HKEY_LOCAL_MACHINE\SOFTWARE\IBM\MQSeries\CurrentVersion\Configuration\QueueManager\`*qmname*`\` , kde *qmname* je název správce front.

V případě systémů UNIX and Linux se komponenty služeb spouštějí v pořadí, ve kterém jsou definovány v souboru qm.ini . V systému Windows, protože se používá registr Windows, produkt WebSphere MQ vyvolá volání **RegEnumKey** , které vrací hodnoty v abecedním pořadí. Proto jsou služby v systému Windows volány v abecedním pořadí, jak jsou definovány v registru.

Pořadí definic `ServiceComponent` je významné. Toto pořadí určuje pořadí, ve kterém jsou komponenty spuštěny pro danou službu. Například, `AuthorizationService` v systému Windows je konfigurován s výchozí komponentou OAM s názvem `MQSeries.WindowsNT.auth.service`. Další komponenty lze definovat pro tuto službu, aby bylo možné přepsat výchozí hodnotu OAM. Pokud není zadán parametr `MQCACF_SERVICE_COMPONENT` , použije se první komponenta rozpoznala v abecedním pořadí ke zpracování požadavku a použije se název této komponenty.

Návratové kódy

Komponenty služeb poskytují správci front návratové kódy pro vytváření sestav o různých podmínkách. Ohlašují úspěch nebo selhání operace a označují, zda má správce front pokračovat do další komponenty služby. Samostatná hodnota parametru *Continuation* je tato indikace.

Data komponent

Jedna komponenta služby může vyžadovat sdílení dat mezi různými funkcemi. Instalovatelné služby poskytují volitelnou datovou oblast, která má být předána při každém vyvolání komponenty služby. Tato datová oblast je určena pro výlučné použití komponenty služby. Je sdílen všemi vyvoláními určité funkce, i když jsou prováděna z různých adresních prostorů nebo procesů. Je zaručeno, že bude adresovatelný z komponenty služby, kdykoli se zavolá. Je třeba deklarovat velikost této oblasti ve stanze *ServiceComponent* .

Inicializace a ukončení komponent
Použití voleb inicializace a ukončení komponenty.

Při vyvolání rutiny inicializace komponenty musí být volána funkce **MQZEP** správce front pro každý vstupní bod podporovaný danou komponentou. **MQZEP** definuje vstupní bod do služby. Předpokládá se, že všechny nedefinované výstupní body jsou NULL.

Komponenta se vždy vyvolá jednou s primární volbou inicializace, dříve než je vyvolána jiným způsobem.

Komponenta může být vyvolána se sekundární volbou inicializace na určitých platformách. Může být například vyvolán jednou pro každý proces operačního systému, podproces nebo úlohu, ke které je služba přistupována.

Je-li použita sekundární inicializace:

- Komponenta může být vyvolána více než jednou pro sekundární inicializaci. Pro každé takové volání se vydá odpovídající volání pro sekundární ukončení, když již služba není potřebná.

  Pro služby názvů se jedná o volání MQZ_TERM_NAME.

  U autorizačních služeb se jedná o volání MQZ_TERM_AUTHORITY.
- Vstupní body musí být znovu zadány (voláním MQZEP) pokaždé, když je komponenta volána pro primární a sekundární inicializaci.
- Pro komponentu se použije pouze jedna kopie dat komponenty; pro každou sekundární inicializaci není jiná kopie.
- Komponenta není vyvolána pro žádné další volání na službu (z procesu operačního systému, vlákna nebo úlohy, jak je to vhodné) před sekundární inicializací.
- Komponenta musí nastavit parametr *Version* na stejnou hodnotu pro primární a sekundární inicializaci.

Komponenta se vždy vyvolá s jednou volbou primárního ukončení jednou, když již není potřeba. K této komponentě nejsou vytvořena žádná další volání.

Komponenta je vyvolána se sekundární volbou ukončení, pokud byla vyvolána pro sekundární inicializaci.

správce oprávnění k objektu (OAM)
Komponenta autorizační služba dodaná s produkty WebSphere MQ se nazývá OAM (Object Authority Manager).

Ve výchozím nastavení je OAM aktivní a pracuje s řídicími příkazy **dspmqaut** (oprávnění k zobrazení),**dmpmqaut** (oprávnění k výpisu) a **setmqaut** (nastavit nebo resetovat oprávnění).

Syntaxe těchto příkazů a jejich použití jsou popsány v části Řídicí příkazy.

OAM pracuje s *entitou* činitele nebo skupiny.

- Na systémech UNIX and Linux :
  - činitel je ID uživatele nebo ID přidružené k aplikačnímu programu spuštěnému jménem uživatele.

  - skupina je systémem UNIX nebo systémem Linux definovanou kolekcí činitelů.
  - Oprávnění lze udělovat nebo odvolat pouze na úrovni skupiny. Požadavek na udělení nebo odebrání oprávnění uživatele aktualizuje primární skupinu pro tohoto uživatele.
- Na systémech Windows:
  - řídicí služba je ID uživatele systému Windows nebo ID přidružené k aplikačnímu programu spuštěnému jménem uživatele.
  - skupina je skupina Windows.
  - Oprávnění mohou být udělována nebo odvolány na úrovni činitele nebo skupiny.

Je-li vydán požadavek na rozhraní MQI nebo je zadán příkaz, zkontroluje produkt OAM autorizaci entity přidružené k operaci a zjišťuje, zda může provádět následující operace:

- Proveďte požadovanou operaci.
- Přistupte k určeným prostředkům správce front.

Autorizační služba vám umožňuje rozšířit nebo nahradit kontrolu oprávnění poskytovanou pro správce front tím, že zapisujete svou vlastní komponentu autorizační služby.

služba názvů
Služba názvů je instalovatelná služba, která poskytuje podporu správci front za účelem vyhledání názvu správce front, který vlastní uvedenou frontu. Z názvu služby nelze načíst žádné jiné atributy fronty.

Služba názvů umožňuje aplikaci otevřít vzdálenou frontu pro výstup tak, jako by šlo o lokální fronty. Služba názvů není vyvolána pro objekty jiné než fronty.

Poznámka: Vzdálení fronty ***musí*** mít svůj atribut *Scope* nastaven na CELL.

Když aplikace otevře frontu, vyhledá nejprve název fronty v adresáři správce front. Pokud ji nenajde, prohledá tolik služeb názvů, kolik bylo nakonfigurováno, dokud nenajde takový, který rozpoznává název fronty. Pokud žádný z nich nerozpozná jméno, otevře se otevření.

Služba názvů vrací vlastníka správce front pro danou frontu. Správce front bude poté pokračovat s požadavkem MQOPEN, jako kdyby příkaz určil název fronty a správce front v původní žádosti.

Rozhraní NSI (name service interface) je součástí rámce WebSphere MQ .

Jak funguje služba názvů

Pokud definice fronty určuje atribut *Scope* jako správce front, tj. SCOPE (QMGR) v prostředí MQSC, je definice fronty (spolu se všemi atributy fronty) uložena pouze v adresáři správce front. Tuto volbu nelze nahradit instalovatelnou službou.

Pokud definice fronty uvádí atribut *Scope* jako buňku, tj. SCOPE (CELL) v prostředí MQSC, je definice fronty znovu uložena v adresáři správce front spolu se všemi atributy fronty. Název fronty a správce front je však také uložen ve službě názvů. Není-li k dispozici žádná služba, která by mohla tyto informace uložit, nelze definovat frontu s buňkou *Scope* .

Adresář, ve kterém jsou informace uloženy, může být spravován službou, nebo může služba použít základní službu, například adresář LDAP pro tento účel. V obou případech musí být definice uložené v adresáři zachovány i po ukončení komponenty a správce front, dokud nejsou explicitně odstraněny.

Poznámka:

1. Chcete-li odeslat zprávu do definice lokální fronty vzdáleného hostitele (s rozsahem CELL) na jiném správci front v rámci buňky adresáře pojmenovávání, je třeba definovat kanál.
2. Zprávy nelze načíst přímo ze vzdálené fronty, a to ani v případě, že má rozsah CELL.
3. Při odesílání do fronty s rozsahem CELL není vyžadována žádná definice vzdálené fronty.
4. Služba názvů centrálně definuje cílovou frontu, ačkoli stále ještě potřebujete přenosovou frontu k cílovému správci front a dvojici definic kanálů. Kromě toho musí mít přenosová fronta v lokálním systému stejný název jako správce front, který vlastní cílovou frontu, s oborem buňky, ve vzdáleném systému.

Pokud má například vzdálený správce front název QM01, přenosová fronta v lokálním systému musí mít také název QM01.

Rozhraní autorizační služby
Autorizační služba poskytuje vstupní body pro použití správcem front.

Vstupní body jsou následující:

MQZ_AUTHENTICATE_USER
Ověřuje ID uživatele a heslo a může nastavit pole kontextu identity.

MQZ_CHECK_AUTHORITY
Kontroluje, zda má entita oprávnění provést jednu nebo více operací na uvedeném objektu.

MQZ_CHECK_PRIVILEGED
Kontroluje, zda je určený uživatel privilegovaným uživatelem.

MQZ_COPY_ALL_AUTHORITY
Kopíruje všechna aktuální oprávnění, která existují pro odkazovaný objekt, na jiný objekt.

OPRÁVNĚNÍ MQZ_DELETE_AUTHORITY
Odstraní všechny autorizace přidružené k uvedenému objektu.

MQZ_ENUMERATE_AUTHORITY_DATA
Načte všechna data oprávnění, která se shodují s uvedenými kritérii výběru.

MQZ_FREE_USER
Uvolní přidružené přidělené prostředky.

FUNKCE MQZ_GET_AUTHORITY
Získá oprávnění, které má entita pro přístup k uvedenému objektu.

MQZ_GET_EXPLICITNÍ_AUTORITA
Získá buď oprávnění, které má pojmenovaná skupina k přístupu k uvedenému objektu (ale bez dalšího oprávnění skupiny **nikdo**), nebo oprávnění, které má primární skupina uvedeného hlavního objektu k přístupu k uvedenému objektu.

MQZ_INIT_AUTHORITY
Inicializuje komponentu autorizační služby.

MQZ_DOTÁZAT SE
Dotáže se podporované funkčnosti autorizační služby.

MQZ_REFRESH_CACHE
Aktualizujte všechny autorizace.

OPRÁVNĚNÍ MQZ_SET_AUTHORITY
Nastaví oprávnění, které má entita k uvedenému objektu.

OPRÁVNĚNÍ MQZ_TERM_AUTHORITY
Ukončí komponentu autorizační služby.

Kromě toho v produktu WebSphere MQ for Windowsautorizační služba poskytuje 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 systému Windows (NT SID).

Tyto názvy jsou definovány jako **typedef**v hlavičkovém souboru `cmqzc.h`, který lze použít k vytvoření prototypu funkcí komponent.

Inicializační funkce (**MQZ_INIT_AUTHORITY**) musí být hlavním vstupním bodem komponenty. Ostatní funkce jsou vyvolány přes adresu vstupního bodu, kterou inicializační funkce přidala do vektoru vstupního bodu komponenty.

Rozhraní služby názvů
Služba názvů poskytuje vstupní body pro použití správcem front.

K dispozici jsou následující vstupní body:

NÁZEV MQZ_INIT_NAME
Inicializovat komponentu služby názvů.

NÁZEV MQZ_TERM_NAME
Ukončete komponentu služby názvů.

NÁZEV MQZ_LOOKUP_NAME
Vyhledejte název správce front pro danou frontu.

MQZ_INSERT_NAME
Vložte položku obsahující název správce front vlastníka pro určenou frontu do adresáře používaného touto službou.

MQZ_DELETE_NÁZEV
Vymažte záznam pro uvedenou frontu z adresáře používaného službou.

Je-li nakonfigurována více než jedna služba názvů:

- Pro vyhledávání je vyvolána funkce MQZ_LOOKUP_NAME pro každou službu v seznamu, dokud nebude název fronty vyřešen (pokud některá komponenta neoznačuje, že by mělo hledání zastavit).
- Pro vložení je vyvolána funkce MQZ_INSERT_NAME pro první službu v seznamu, která podporuje tuto funkci.
- Pro odstranění je funkce MQZ_DELETE_NAME vyvolána pro první službu v seznamu, která podporuje tuto funkci.

Nemít více než jednu komponentu, která podporuje funkce vložení a odstranění. Avšak komponenta, která podporuje pouze vyhledávání, je proveditelná a lze ji použít například jako poslední komponentu v seznamu k vyřešení jakéhokoli názvu, který není znám žádnou jinou komponentou služby názvů, ke správci front, ve kterém lze definovat název.

V programovacím jazyku C jsou názvy definovány jako datové typy funkcí pomocí příkazu typedef. Lze je použít k vytvoření prototypu servisních funkcí, aby se zajistilo, že parametry jsou správné.

Soubor záhlaví, který obsahuje veškerý materiál specifický pro instalovatelné služby, je `cmqzc.h` pro jazyk C.

Kromě inicializační funkce (MQZ_INIT_NAME), která musí být hlavním vstupním bodem komponenty, jsou funkce vyvolány adresou vstupního bodu, kterou funkce inicializace přidala, pomocí volání MQZEP.

Použití více komponent služeb
Pro službu můžete instalovat více než jednu komponentu. To umožňuje komponentám poskytovat pouze částečné implementace služby a spolehnout se na ostatní komponenty, aby poskytly zbývající funkce.

Příklad použití více komponent

Předpokládejme, že vytvoříte dvě komponenty služby názvů nazvané `ABC_name_serv` a `XYZ_name_serv`.

`ABC_name_serv`
Tato komponenta podporuje vložení názvu do adresáře služeb nebo odstranění jeho názvu z adresáře služeb, ale nepodporuje vyhledávání názvu fronty.

`XYZ_name_serv`
Tato komponenta podporuje vyhledání názvu fronty, ale nepodporuje vložení názvu do adresáře služeb nebo odstranění názvu z adresáře služeb.

Komponenta `ABC_name_serv` uchovává databázi názvů front a používá dva jednoduché algoritmy k vložení nebo odstranění názvu z adresáře služby.

Komponenta XYZ_name_serv používá jednoduchý algoritmus, který vrací pevný název správce front pro všechny názvy front, s nimiž je vyvolána. Neobsahuje databázi názvů front, a proto nepodporuje funkce vkládání a odstraňování.

Komponenty jsou instalovány ve stejném správci front. Stanzy *ServiceComponent* jsou seřazeny tak, že komponenta ABC_name_serv je vyvolána jako první. Jakákoli volání pro vložení nebo odstranění fronty v adresáři komponenty jsou ošetřena komponentou ABC_name_serv; je to jediná, která implementuje tyto funkce. Avšak volání vyhledání, které komponenta ABC_name_serv nemůže interpretovat, je předáno do komponenty pouze pro vyhledávání, XYZ_name_serv. Tato komponenta dodává název správce front z jednoduchého algoritmu.

Vynechání vstupních bodů při použití více komponent

Pokud se rozhodnete použít více komponent k poskytování služby, můžete navrhnout komponentu služby, která neimplementuje určité funkce. Rámec instalovatelných služeb neklade žádná omezení, na které můžete vynechat. Avšak v případě určitých instalovatelných služeb může být vynechání jedné nebo více funkcí logicky nekonzistentní s účelem služby.

Příklad vstupních bodů použitých s více komponentami

Tabulka 53 na stránce 368 ukazuje příklad instalovatelné služby názvů, pro kterou byly nainstalovány dvě komponenty. Každá podporuje jinou sadu funkcí přidružených k této konkrétní instalovatelné službě. Pro funkci vložení je nejprve vyvolán vstupní bod komponenty ABC. Vstupní body, které nebyly definovány pro službu (pomocí **MQZEP**), jsou považovány za NULL. Vstupní bod pro inicializaci je uveden v tabulce, ale to není povinné, protože inicializace je prováděna hlavním vstupním bodem komponenty.

Má-li správce front použít instalovatelnou službu, použije vstupní body definované pro tuto službu (sloupce v produktu Tabulka 53 na stránce 368). Při použití každé komponenty správce front určuje adresu rutiny, která implementuje požadovanou funkci. To pak volá rutinu, pokud existuje. Je-li operace úspěšná, správce front použije všechny výsledky a informace o stavu.

Tabulka 53. Příklad vstupních bodů pro instalovatelnou službu

Číslo funkce	**Komponenta služby názvu ABC**	**Komponenta služby názvů XYZ**
MQZID_INIT_NAME (Inicializace)	Funkce ABC_initialize ()	XYZ_initialize ()
MQZID_TERM_NAME (Ukončení)	Funkce ABC_terminate ()	XYZ_terminate ()
MQZID_INSERT_NAME (Vložení)	Funkce ABC_Insert ()	NULL
MQZID_DELETE_NAME (Výmaz)	Funkce ABC_Delete ()	NULL
MQZID_LOOKUP_NAME (vyhledání)	NULL	XYZ_Vyhledávání ()

Pokud rutina neexistuje, bude tento proces ve správci front opakován pro další komponentu v seznamu. Kromě toho, pokud rutina existuje, ale vrací kód označující, že nemohl operaci provést, pokračuje pokus s další dostupnou komponentou. Rutiny v komponentách služeb mohou vrátit kód, který označuje, že by se neměly provádět žádné další pokusy o provedení operace.

Konfigurace služeb a komponent

Konfigurujte komponenty služeb pomocí konfiguračních souborů správce front, kromě systémů Windows , kde má každý správce front v registru svou vlastní sekci.

1. Chcete-li definovat službu pro správce front a určit umístění modulu, přidejte do konfiguračního souboru správce front oddíly.

   Každá použitá služba musí mít objekt stanza *Service* , který definuje službu pro správce front.

   Pro každou komponentu v rámci služby musí existovat stanza *ServiceComponent* . Identifikuje název a cestu k modulu, který obsahuje kód dané komponenty.

Další informace viz “Formát sekce služby” na stránce 369 a “Formát sekce komponent služby” na stránce 369

Komponenta autorizační služba, známá jako OAM (Object Authority Manager), se dodává spolu s produktem. Při vytváření správce front je konfigurační soubor správce front (nebo registr v systému Windows) automaticky aktualizován tak, aby obsahoval příslušné oddíly pro autorizační službu a pro výchozí komponentu (OAM). Pro ostatní komponenty musíte nakonfigurovat konfigurační soubor správce front ručně.

Kód pro každou komponentu služby je načten do správce front při spuštění správce front s použitím dynamické vazby, kde je tato podpora na platformě podporována.

2. Chcete-li aktivovat komponentu, zastavte a znovu spusťte správce front.

Formát sekce služby
Sekce Service obsahuje název služby a počet vstupních bodů definovaných pro službu.

Formát stanzy je následující:

```
Service:
  Name=<service_name>
  EntryPoints=<entries>
```

kde:

<service_name>
Název služby. Toto je definováno službou.

<entries>
Počet vstupních bodů definovaných pro službu. To zahrnuje vstupní body inicializace a ukončení.

Formát služby stanza pro systémy Windows
Na systémech Windows obsahuje stanza *Service* atribut SecurityPolicy .

Formát objektu stanza je:

```
Service:
  Name=<service_name>
  EntryPoints=<entries>
  SecurityPolicy=<policy>
```

kde:

<service_name>
Název služby. Toto je definováno službou.

<entries>
Počet vstupních bodů definovaných pro službu. To zahrnuje vstupní body inicializace a ukončení.

<policy>
NTSIDsRequired (Identifikátor zabezpečení systému Windows) nebo Default. Pokud nezadáte NTSIDsRequired, použije se hodnota Default . Tento atribut je platný pouze tehdy, má-li Name hodnotu AuthorizationService.

Další informace najdete v tématu “Konfigurace oddílů autorizační služby: systémy Windows” na stránce 370.

Formát sekce komponent služby

Formát stanzy komponenty služby je:

```
ServiceComponent:
  Service=<service_name>
  Name=<component_name>
  Module=<module_name>
  ComponentDataSize=<size>
```

kde:

<service_name>
Název služby. Musí odpovídat Name uvedenému ve stanze služby.

<component_name>
Popisný název komponenty služby. Musí být jedinečný a obsahovat pouze znaky platné pro názvy objektů produktu WebSphere MQ (například názvy front). Tento název se vyskytuje ve zprávách operátora generovaných službou. Doporučujeme použít jméno začínající ochrannou známkou společnosti nebo obdobným rozlišovacím řetězcem.

<module_name>
Název modulu, který má obsahovat kód pro tuto komponentu.

<size>
Velikost oblasti dat komponenty předané komponentě při každém volání v bajtech. Uveďte nulu, pokud nejsou požadována žádná data komponenty.

Tyto dvě stanzy se mohou vyskytnout v libovolném pořadí a klíče stanzy pod nimi se mohou objevit také v libovolném pořadí. Pro každou z těchto stanz musí být všechny klíče oddílu přítomny. Je-li klíč oddílu duplikován, použije se poslední.

Při spuštění správce front zpracuje všechny položky komponenty služby v konfiguračním souboru postupně. Poté načte uvedený modul komponenty, vyvolá vstupní bod komponenty (která musí být vstupním bodem pro inicializaci komponenty) a předá ji obslužnou rutinu konfigurace.

Konfigurace stanzy autorizační služby: systémy UNIX and Linux
V systémech UNIX and Linux má každý správce front svůj vlastní konfigurační soubor správce front.

Například výchozí cesta a název souboru s konfiguračním souborem správce front pro správce front QMNAME je `/var/mqm/qmgrs/QMNAME/qm.ini`.

Stanza *`Service`* a stanza *`ServiceComponent`* pro výchozí autorizační komponentu se přidávají do `qm.ini` automaticky, ale mohou být přepsány `mqsnoaut`. Jakékoli další oddíly *`ServiceComponent`* musí být přidány ručně.

Například následující sekce v konfiguračním souboru správce front definují dvě komponenty autorizační služby v produktu WebSphere MQ for AIX. *`MQ_INSTALLATION_PATH`* představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

```
Service:
   Name=AuthorizationService
   EntryPoints=13

ServiceComponent:
   Service=AuthorizationService
   Name=MQSeries.UNIX.auth.service
   Module=MQ_INSTALLATION_PATH/lib/amqzfu
   ComponentDataSize=0

ServiceComponent:
   Service=AuthorizationService
   Name=user.defined.authorization.service
   Module=/usr/bin/udas01
   ComponentDataSize=96
```

Obrázek 73. Stanzy autorizační služby produktu UNIX and Linux v souboru qm.ini

Stanza service component (`MQSeries.UNIX.auth.service`) definuje výchozí komponentu autorizační služby, OAM. Pokud odeberete tuto stanzu a restartujete správce front, bude OAM zablokováno a nebudou provedeny žádné kontroly autorizace.

Konfigurace oddílů autorizační služby: systémy Windows
V produktu WebSphere MQ for Windows má každý správce front v registru svou vlastní sekci.

Stanza *Service* a stanza *ServiceComponent* pro výchozí autorizační komponentu jsou přidány do registru automaticky, ale mohou být přepsány pomocí `mqsnoaut`. Jakékoli další oddíly *ServiceComponent* musí být přidány ručně.

Atribut `SecurityPolicy` můžete také přidat pomocí služeb produktu WebSphere MQ . Atribut `SsecurityPolicy` se používá pouze v případě, že služba uvedená ve stanze *Service* je autorizační služba, to znamená výchozí OAM. Atribut `SecurityPolicy` vám umožňuje uvést zásady zabezpečení pro každého správce front. Možné hodnoty jsou:

Default
Uveďte `Default` , pokud chcete, aby se projevila výchozí zásada zabezpečení. Pokud identifikátor zabezpečení systému Windows (NT SID) není předán OAM pro konkrétní ID uživatele, provede se pokus o získání příslušného SID prohledáváním příslušných databází zabezpečení.

NTSIDsRequired
Vyžaduje, aby při provádění kontrol zabezpečení byl při provádění kontroly zabezpečení předán identifikátor SID systému NT.

Informace o formátu sekce Service stanza viz “Formát služby stanza pro systémy Windows” na stránce 369. Další obecné informace o zabezpečení najdete v tématu Nastavení zabezpečení v systémech Windows, UNIX and Linux.

Stanza Service Component, `MQSeries.WindowsNT.auth.service` definuje výchozí komponentu autorizační služby, OAM. Pokud odeberete tuto stanzu a restartujete správce front, bude OAM zablokováno a nebudou provedeny žádné kontroly autorizace.

Konfigurace oddílů služby názvů: Systémy Unix a Linux
Sem zadejte krátký popis; použije se pro první odstavec a abstrakt.

Následující příklady stanz konfiguračního souboru UNIX and Linux pro službu názvů určují komponentu služby názvů, kterou poskytla (fiktivní) společnost ABC.

```
# Stanza for name service
 Service:
    Name=NameService
    EntryPoints=5

# Stanza for name service component, provided by ABC
 ServiceComponent:
    Service=NameService
    Name=ABC.Name.Service
    Module=/usr/lib/abcname
    ComponentDataSize=1024
```

Obrázek 74. Stanzy služby názvů v souboru qm.ini (pro systémy UNIX and Linux)

Poznámka: Na systémech Windows je informace o objektu stanza názvu uložena v registru.

Aktualizace OAM po změně autorizace uživatele
V produktu WebSphere MQmůžete aktualizovat informace o skupině autorizace OAM ihned po změně členství skupiny autorizace uživatele, což odráží změny provedené na úrovni operačního systému, aniž by bylo nutné zastavit a znovu spustit správce front. Chcete-li to provést, zadejte příkaz **REFRESH SECURITY** .

Poznámka: Když změníte autorizace pomocí příkazu `setmqaut` , OAM tyto změny okamžitě naimplementuje.

Správci front ukládají autorizační data do lokální fronty s názvem SYSTEM.AUTH.DATA.QUEUE. Tato data jsou spravována pomocí `amqzfuma.exe` .

Související odkazy
REFRESH SECURITY

Zápis a kompilace uživatelských procedur rozhraní API

Uživatelské procedury rozhraní API vám umožňují psát kód, který změní chování volání rozhraní API produktu WebSphere MQ , jako je například MQPUT a MQGET, a pak tento kód vloží bezprostředně před nebo bezprostředně po těchto voláních.

Poznámka: Nepodporováno na produktu WebSphere MQ pro z/OS.

Proč používat uživatelské procedury rozhraní API?

Každá z vašich aplikací má specifickou úlohu a její kód by měl provést tuto úlohu co nejefektivněji. Na vyšší úrovni byste mohli chtít použít standardy nebo obchodní procesy pro konkrétního správce front pro **všechny** aplikace, které tento správce front používají. Je účinnější provádět tuto úroveň nad úrovní jednotlivých aplikací, a tudíž bez nutnosti měnit kód každé ovlivněné aplikace.

Zde je několik návrhů oblastí, ve kterých mohou být uživatelské procedury rozhraní API užitečné:

- V případě *zabezpečení*můžete poskytnout ověření a zkontrolovat, zda jsou aplikace autorizovány pro přístup ke správci front nebo ke správci front. Můžete také použít rozhraní API pro práci s policejními aplikacemi, ověřovat jednotlivá volání rozhraní API nebo dokonce i parametry, které používají.
- Pro *flexibilitu*můžete reagovat na rychlé změny ve vašem obchodním prostředí, aniž byste změnili aplikace, které spoléhají na data v daném prostředí. Mohli byste například mít uživatelské procedury rozhraní API, které reagují na změny úrokových sazeb, směnných kurzů měn nebo ceny komponent ve výrobním prostředí.
- Pro *monitorování* použití fronty nebo správce front můžete trasovat tok aplikací a zpráv, chyby v protokolu v voláních rozhraní API, nastavit záznamy monitorování pro účely evidence nebo shromažďovat statistiky využití pro účely plánování.

Co se stane, když se spustí uživatelská procedura rozhraní API?

Jakmile jste napsali uživatelský program a identifikovali jej do produktu WebSphere MQ, správce front automaticky vyvolá váš výstupní kód v registrovaných bodech.

Rutiny ukončení rozhraní API, které se mají spustit, jsou identifikovány ve stanzách v systémech IBM i, Windows, UNIX and Linux . Toto téma se vztahuje na sekce v konfiguračních souborech mqs.ini a qm.ini.

Definice rutin se může vyskytnout na třech místech:

1. ApiExitCommon, v souboru mqs.ini , identifikuje rutiny pro celý produkt WebSphere MQ, který se použije při spuštění správců front. Ty mohou být přepsány rutinami definovanými pro jednotlivé správce front (viz položka “3” na stránce 372 v tomto seznamu).
2. Šablona ApiExitv souboru mqs.ini identifikuje rutiny pro celou sadu WebSphere MQ, která byla zkopírována do lokální sady ApiExit(viz položka “3” na stránce 372 v tomto seznamu) při vytvoření nového správce front.
3. ApiExitLokální, v souboru qm.ini , identifikuje rutiny, které se vztahují na konkrétního správce front.

Při vytvoření nového správce front se definice šablon ApiExitv souboru mqs.ini zkopírují do lokálních definic ApiExitv souboru qm.ini nového správce front. Když je spuštěn správce front, jsou použity jak lokální definice ApiExit, tak i lokální definice ApiExit. Lokální definice ApiExitnahrazují obecné definice ApiExit, pokud obě identifikují rutinu se stejným názvem. Atribut `Sequence` , který je popsán v “Konfigurace uživatelských procedur rozhraní API” na stránce 378 , určuje pořadí, ve kterém jsou rutiny definované ve stanzách spuštěny.

Použití uživatelských procedur rozhraní API ve více instalacích produktu WebSphere MQ

Ujistěte se, že uživatelské procedury rozhraní API napsané pro dřívější verzi produktu WebSphere MQ se používají pro práci se všemi verzemi, protože změny provedené ve verzi 7.1 nemusí pracovat se starší verzí. Další informace o změnách provedených pro ukončení naleznete v tématu “Uživatelské procedury pro psaní a kompilaci a instalovatelné služby” na stránce 358.

Ukázky poskytnuté pro uživatelské procedury rozhraní API amqsaem a amqsaxe odrážejí změny vyžadované při zápisu uživatelských procedur. Aplikace klienta musí zajistit, aby byly před spuštěním aplikace propojeny správné knihovny produktu WebSphere MQ , které odpovídají instalaci správce front, k němuž je aplikace přidružena, do ní souvisí.

Zápis uživatelských procedur API

Uživatelské procedury můžete zapsat pro každé volání API pomocí programovacího jazyka C.

Uživatelské procedury jsou k dispozici pro každé volání rozhraní API takto:

- MQCB, chcete-li znovu registrovat zpětné volání pro zadaný popisovač objektu a řídicí aktivaci a změny pro zpětné volání
- MQCTL, k provedení řídicích akcí na manipulátorech objektů otevřených pro připojení
- MQCONN/MQCONNX, který poskytuje manipulátor připojení správce front pro použití při následných voláních rozhraní API
- MQDISC, k odpojení od správce front
- MQBEGIN, chcete-li zahájit globální pracovní jednotku (UOW)
- MQBACK, chcete-li zálohovat jednotku UOW
- MQCMIT, k potvrzení jednotky UOW
- MQOPEN, chcete-li otevřít prostředek WebSphere MQ pro následný přístup
- MQCLOSE, chcete-li zavřít prostředek WebSphere MQ , který byl dříve otevřen pro přístup
- MQGET, chcete-li načíst zprávu z fronty, která byla dříve otevřena pro přístup
- MQPUT1, chcete-li umístit zprávu do fronty
- MQPUT, chcete-li umístit zprávu do fronty, která byla dříve otevřena pro přístup
- MQINQ-zjišťování atributů prostředku WebSphere MQ , který byl dříve otevřen pro přístup
- MQSET, chcete-li nastavit atributy fronty, která byla dříve otevřena pro přístup
- MQSTAT, chcete-li načíst informace o stavu
- MQSUB, chcete-li registrovat odběr aplikací pro konkrétní téma
- MQSUBRQ, chcete-li provést požadavek na odběr

MQ_CALLBACK_EXIT poskytuje funkci ukončení, která se má provést před a po zpracování zpětného volání. Další informace viz Callback-MQ_CALLBACK_EXIT.

V rámci uživatelských procedur rozhraní API má volání obecnou podobu:

```
MQ_call_EXIT (parameters, context, ApiCallParameters)
```

kde *call* je název volání MQI bez předpony MQ ; např. PUT, GET. Ovládací prvek *parameters* řídí funkci uživatelské procedury, která primárně poskytuje komunikaci mezi ukončením a externím řídicím blokům MQAXP (struktura výstupních parametrů rozhraní API) a MQAXC (struktura kontextu uživatelské procedury rozhraní API). *context* popisuje kontext, ve kterém byla volána uživatelská procedura rozhraní API, a *ApiCallParameters* představují parametry pro volání MQI.

Při zápisu uživatelské procedury rozhraní API je k dispozici ukázkový výstup amqsaxe0.c; tento výstup generuje trasovací záznamy do souboru, který jste zadali. Tuto ukázku můžete použít jako výchozí bod při zápisu uživatelských procedur. Další informace o použití ukázkové uživatelské procedury naleznete v tématu “Ukázkový program uživatelské procedury rozhraní API” na stránce 108.

Další informace o voláních uživatelské procedury rozhraní API, externích řídicích blocích a přidružených tématech naleznete v tématu Odkaz na ukončení rozhraní API.

Obecné informace o tom, jak zapisovat, kompilovat a konfigurovat ukončení, naleznete v části “Uživatelské procedury pro psaní a kompilaci a instalovatelné služby” na stránce 358.

Používání popisovačů zpráv v uživatelských procedurách rozhraní API

Můžete řídit, ke kterým vlastnostem zprávy má přístup rozhraní API přístup. Vlastnosti jsou přidruženy k manipulátoru ExitMsg. Vlastnosti nastavené ve výstupní frontě jsou nastaveny na vkládanou zprávu, ale vlastnosti načtené v rámci procedury get exit se do aplikace nevrátí.

Pokud registrujete uživatelskou funkci MQ_INIT_EXIT pomocí volání MQXEP MQI s parametrem **Function** nastaveným na hodnotu MQXF_INIT a **ExitReason** nastaveným na hodnotu MQXR_CONNECTION, předáváte strukturu MQXEPO jako parametr **ExitOpts** . Struktura MQXEPO obsahuje pole ExitProperties , které určuje sadu vlastností, které mají být zpřístupněny pro ukončení. Je zadán jako znakový řetězec reprezentující předponu vlastností, která odpovídá názvu složky MQRFH2 .

Každá uživatelská procedura rozhraní API přijímá strukturu MQAXP obsahující pole manipulátoru ExitMsg. Toto pole je nastaveno na hodnotu vygenerovanou produktem WebSphere MQ a je specifická pro připojení. Popisovač je tedy nezměněn mezi uživatelskými procedurami rozhraní API stejného nebo různých typů ve stejném připojení.

V příkazu MQ_PUT_EXIT nebo MQ_PUT1_EXIT s **ExitReason** MQXR_BEFORE, to znamená ukončení rozhraní API před vložením zprávy, jakékoli vlastnosti (jiné než vlastnosti deskriptoru zpráv) přidružené k obslužné rutiny ExitMsg, když je dokončení uživatelské procedury nastaveno na vkládané zprávy. Chcete-li tomu zabránit, nastavte ovladač ExitMsgna hodnotu MQHM_NONE. Můžete také dodat jiný popisovač zprávy.

Při registraci MQ_GET_EXIT je popisovač ExitMsgvymazán z vlastností a je naplněn vlastnostmi uvedenými v poli ExitProperties , když byla registrována hodnota MQ_INIT_EXIT, kromě vlastností deskriptoru zpráv. Tyto vlastnosti nejsou k dispozici pro získání aplikace. Je-li aplikace pro získání zprávy v poli MQGMO (Získat volby zprávy) zadána, jsou pro uživatelskou proceduru rozhraní API k dispozici všechny vlastnosti přidružené k tomuto popisovači včetně vlastností deskriptoru zpráv. Chcete-li zabránit tomu, aby byl popisovač ExitMsgnaplněn vlastnostmi, nastavte jej na hodnotu MQHM_NONE.

K dispozici je ukázkový program amqsaem0.c, který ilustruje použití obslužných rutin zpráv v uživatelských procedurách rozhraní API.

Kompilace uživatelských procedur rozhraní API

Po zápisu uživatelské procedury zkompilujete a propojíte jej následujícím způsobem.

Následující příklady zobrazují příkazy použité pro ukázkový program popsaný v části “Ukázkový program uživatelské procedury rozhraní API” na stránce 108. Pro jiné platformy než systémy Windows můžete najít vzorový kód ukončení rozhraní API v produktu *MQ_INSTALLATION_PATH*/samp a v *MQ_INSTALLATION_PATH*/samp/binv kompilované a propojené sdílené knihovně. Na systémech Windows můžete najít vzorový kód ukončení rozhraní API v produktu *MQ_INSTALLATION_PATH*\Tools\c\Samples. *MQ_INSTALLATION_PATH* Představuje adresář, ve kterém byl nainstalován produkt WebSphere MQ .

Poznámka pro uživatele:

1. Pokyny pro programování 64bitových aplikací jsou uvedeny v tématu Kódování standardů na 64bitových platformách

V případě zavedení klientů výběrového vysílání je možné na straně klienta spustit uživatelské procedury rozhraní API a uživatelské procedury pro převod dat, protože některé zprávy nemusí procházet správcem front. Následující knihovny jsou nyní součástí balíků klienta stejně jako balíky serveru:

Tabulka 54. Knihovny, které jsou nyní v balících klienta a serveru

Operační systém	**Knihovny**
Windows	32 bit & 64 bit: mqm.dll & mqm.pdb
Linux & HP-UX	32 bit & 64 bit: libmqm.so & libmqm_r.so
AIX	32 bit & 64 bit: libmqm.a & libmqm_r.a
Solaris	32 bitů & 64 bitů: libmqm.so

Kompilace uživatelských procedur rozhraní API na systémech Unix a Linux
Příklady způsobu kompilace uživatelských procedur rozhraní API na systémech UNIX a Linux .

Na všech platformách je vstupní bod do modulu MQStart.

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

V systému AIX

Zkompilujte zdrojový kód uživatelské procedury rozhraní API zadáním jednoho z následujících příkazů:

32bitové aplikace

Nevláknová

```
cc -e MQStart -bE:amqsaxe.exp -bM:SRE -o /var/mqm/exits/amqsaxe \
    amqsaxe0.c -IMQ_INSTALLATION_PATH/inc
```

Vláknové

```
xlc_r -e MQStart -bE:amqsaxe.exp -bM:SRE -o /var/mqm/exits/amqsaxe_r \
    amqsaxe0.c -IMQ_INSTALLATION_PATH/inc
```

64bitové aplikace

Nevláknová

```
cc -q64 -e MQStart -bE:amqsaxe.exp -bM:SRE -o /var/mqm/exits64/amqsaxe \
    amqsaxe0.c -IMQ_INSTALLATION_PATH/inc
```

Vláknové

```
xlc_r -q64 -e MQStart -bE:amqsaxe.exp -bM:SRE -o /var/mqm/exits64/amqsaxe_r \
    amqsaxe0.c -IMQ_INSTALLATION_PATH/inc
```

Na platformě HP-UX Itanium

32bitové aplikace

Nevláknová

Kompilace zdrojového kódu ukončení API:

```
c89 +e +z -c -D_HPUX_SOURCE -o amqsaxe.o amqsaxe0.c -IMQ_INSTALLATION_PATH/inc
```

Odkaz na zdrojový kód uživatelské procedury rozhraní API

```
ld +b: -b amqsaxe.o +ee MQStart -o /var/mqm/exits/amqsaxe
rm amqsaxe.o
```

Vláknové

Kompilace zdrojového kódu ukončení API:

```
c89 +e +z -c -D_HPUX_SOURCE -o amqsaxe.o amqsaxe0.c -IMQ_INSTALLATION_PATH/inc
```

Odkaz na zdrojový kód uživatelské procedury rozhraní API

```
ld +b: -b amqsaxe.o +ee MQStart -o /var/mqm/exits/amqsaxe_r
rm amqsaxe.o
```

64bitové aplikace

Nevláknová

Kompilace zdrojového kódu ukončení API:

```
c89 +DD64 +e +z -c -D_HPUX_SOURCE -o amqsaxe.o amqsaxe0.c -IMQ_INSTALLATION_PATH/inc
```

Odkaz na zdrojový kód uživatelské procedury rozhraní API

```
ld -b amqsaxe.o +ee MQStart -o  /var/mqm/exits64/amqsaxe
rm amqsaxe.o
```

Vláknové

Kompilace zdrojového kódu ukončení API:

```
c89 +DD64 +e +z -c -D_HPUX_SOURCE -o amqsaxe.o amqsaxe0.c -IMQ_INSTALLATION_PATH/inc
```

Odkaz na zdrojový kód uživatelské procedury rozhraní API

```
ld -b amqsaxe.o +ee MQStart -o  /var/mqm/exits64/amqsaxe_r
rm amqsaxe.o
```

zapLinux

Zkompilujte zdrojový kód uživatelské procedury rozhraní API zadáním jednoho z následujících příkazů:

31bitové aplikace

Nevláknová

```
 gcc -m31 -shared -fPIC -o /var/mqm/exits/amqsaxe amqsaxe0.c \
     -IMQ_INSTALLATION_PATH/inc
```

Vláknové

```
gcc -m31 -shared -fPIC -o /var/mqm/exits/amqsaxe_r amqsaxe0.c \
    -IMQ_INSTALLATION_PATH/inc
```

32bitové aplikace

Nevláknová

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/amqsaxe amqsaxe0.c \
     -IMQ_INSTALLATION_PATH/inc
```

Vláknové

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/amqsaxe_r amqsaxe0.c \
    -IMQ_INSTALLATION_PATH/inc
```

64bitové aplikace

Nevláknová

```
 gcc -m64 -shared -fPIC -o /var/mqm/exits64/amqsaxe amqsaxe0.c \
     -IMQ_INSTALLATION_PATH/inc
```

Vláknové

```
 gcc -m64 -shared -fPIC -o /var/mqm/exits64/amqsaxe_r amqsaxe0.c \
     -IMQ_INSTALLATION_PATH/inc
```

V systému Solaris

Zkompilujte zdrojový kód uživatelské procedury rozhraní API zadáním jednoho z následujících příkazů:

32bitové aplikace

Platforma SPARC

```
cc -xarch=v8plus -KPIC -mt -G -o /var/mqm/exits/amqsaxe \
   amqsaxe0.c -IMQ_INSTALLATION_PATH/inc \
   -R/usr/lib/32 -lsocket -lnsl -ldl
```

Platformax86-64

```
cc -xarch=386 -KPIC -mt -G -o /var/mqm/exits/amqsaxe \
   amqsaxe0.c -IMQ_INSTALLATION_PATH/inc \
   -R/usr/lib/32 -lsocket -lnsl -ldl
```

64bitové aplikace

Platforma SPARC

```
cc -xarch=v9 -KPIC -mt -G -o /var/mqm/exits64/amqsaxe \
   amqsaxe0.c -IMQ_INSTALLATION_PATH/inc \
   -R/usr/lib/64 -lsocket -lnsl -ldl
```

Platformax86-64

```
cc -xarch=amd64 -KPIC -mt -G -o /var/mqm/exits64/amqsaxe \
   amqsaxe0.c -IMQ_INSTALLATION_PATH/inc \
   -R/usr/lib/64 -lsocket -lnsl -ldl
```

Na systémech Windows

Kompilace a propojení ukázkového ukončovacího programu rozhraní API produktu `amqsaxe0.c`na systému Windows

Soubor typu manifest je volitelný dokument XML obsahující verzi nebo jinou informaci, která může být vložena do kompilované aplikace nebo knihovny DLL.

Pokud takový dokument nemáte, vynechte parametr -manifest *manifest.file* v příkazu **mt** .

Přizpůsobte příkazy v příkladech v produktu Obrázek 75 na stránce 377 nebo Obrázek 76 na stránce 378 za účelem kompilace a propojení `amqsaxe0.c` v systému Windows . Tyto příkazy pracují s produktem Microsoft Visual Studio 2005, 2008 nebo 2010. Příklady předpokládají, že adresář produktu WebSphere MQ `C:\Program Files\IBM\WebSphere 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 75. Kompilace a odkaz `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 76. Kompilace a propojení amqsaxe0.c na 64bitovém systému Windows

Související pojmy

“Ukázkový program uživatelské procedury rozhraní API” na stránce 108
Ukázková uživatelská procedura rozhraní API vygeneruje trasování MQI do uživatelem určeného souboru s předponou definovanou v proměnné prostředí MQAPI_TRACE_LOGFILE.

Konfigurace uživatelských procedur rozhraní API

Chcete-li povolit ukončení rozhraní API tím, že změníte informace o konfiguraci, nakonfigurujte produkt IBM WebSphere MQ .

Chcete-li změnit informace o konfiguraci, musíte změnit stanzy, které definují uživatelské rutiny a pořadí, v jakém se spouštějí. Tyto informace lze změnit následujícími způsoby:

- Použití produktu IBM WebSphere MQ Explorer (na platformách Windows a Linux (platformyx86 a x86-64))
- Použití příkazu **amqmdain** (na systému Windows)
- Přímé použití souborů mqs.ini a qm.ini (na systémech Windows, UNIX and Linux).

  Soubor mqs.ini obsahuje informace vztahující se ke všem správcům front v konkrétním uzlu. Můžete jej najít v adresáři `/var/mqm` na UNIX and Linux a v WorkPath uvedeném v klíči `HKLM\SOFTWARE\IBM\WebSphere MQ` na systémech Windows .

  Soubor qm.ini obsahuje informace vztahující se ke specifickému správci front. Pro každého správce front je k dispozici jeden konfigurační soubor správce front, který je umístěn v kořenovém adresáři adresářového stromu obsazeného správcem front. Příklad: Cesta a název konfiguračního souboru pro správce front s názvem QMNAME je:

  Na systémech UNIX and Linux :

  ```
  /var/mqm/qmgrs/QMNAME/qm.ini
  ```

  Na systémech Windows :

  ```
  C:\Program Files\IBM\WebSphere MQ\qmgrs\QMNAME\qm.ini
  ```

  Před úpravou konfiguračního souboru jej zálohujte tak, abyste měli kopii, na kterou se můžete vrátit, pokud k tomu dojde.

  Konfigurační soubory můžete upravit buď:

  – Automaticky pomocí příkazů, které mění konfiguraci správců front v uzlu
  – Ruční při použití standardního textového editoru

  Pokud jste v atributu konfiguračního souboru nastavili nesprávnou hodnotu, hodnota se ignoruje a vydá se zpráva operátora, která daný problém označuje. (Efekt je stejný jako chybějící atribut zcela.)

Stanzy ke konfiguraci

Oddíly, které musí být změněny, jsou následující:

ApiExitCommon

Definováno v souboru mqs.ini a na stránce vlastností produktu IBM WebSphere MQ Explorer na stránce vlastností produktu IBM WebSphere MQ pod položkou Uživatelské procedury.

Při spuštění správce front jsou atributy v této sekci načteny a poté přepsané uživatelskými procedurami rozhraní API definovanými v souboru qm.ini.

ApiExitTemplate

Definováno v souboru mqs.ini a na stránce vlastností produktu IBM WebSphere MQ Explorer na stránce vlastností produktu IBM WebSphere MQ pod položkou Uživatelské procedury.

Je-li vytvořen správce front, budou atributy v této stanze zkopírovány do nově vytvořeného souboru qm.ini v lokální stanze ApiExit.

ApiExitLocal

Definováno v souboru qm.ini a na stránce IBM WebSphere MQ Explorer na stránce vlastností správce front pod položkou Uživatelské procedury.

Při spuštění správce front jsou zde definované uživatelské procedury rozhraní API potlačují výchozí hodnoty definované v souboru mqs.ini.

Atributy pro stanzy

- Pojmenujte uživatelskou proceduru rozhraní API pomocí následujícího atributu:

  Název = ApiExit_name
  Popisný název uživatelské procedury rozhraní API předané do pole Název ExitInfostruktury MQAXP.

  Tento název musí být jedinečný, nesmí být delší než 48 znaků a smí obsahovat pouze platné znaky pro názvy objektů produktu IBM WebSphere MQ (například názvy front).

- Identifikujte modul a vstupní bod kódu ukončení rozhraní API, které se mají spustit pomocí následujících atributů:

  Funkce=název_funkce
  Název vstupního bodu funkce do modulu, který obsahuje kód ukončení rozhraní API. Tento vstupní bod je funkce MQ_INIT_EXIT.

  Délka pole je omezena hodnotou MQ_EXIT_NAME_LENGTH.

  Modul = název_modulu
  Modul obsahující kód ukončení rozhraní API.

  Pokud pole obsahuje název modulu včetně úplné cesty, je použit beze změny.

  Pokud toto pole obsahuje pouze název modulu, je modul umístěn pomocí atributu `ExitsDefaultPath` v souboru `ExitPath` v souboru qm.ini.

  Na platformách, které podporují samostatné knihovny s podporou podprocesů, je třeba do modulu uživatelské procedury rozhraní API poskytovat jak nevláknovou, tak i verzi s podporou podprocesů. Svláknová verze musí mít příponu `_r` . Svlákněná verze stubu aplikace IBM WebSphere MQ implicitně připojuje `_r` k danému názvu modulu před jeho načtením.

  Délka tohoto pole je omezena na maximální délku cesty, kterou platforma podporuje.

- Volitelně předávejte data pomocí uživatelské procedury pomocí následujícího atributu:

  Data=název_dat
  Data, která mají být předána uživatelské proceduře rozhraní API, v poli ExitData struktury MQAXP.

  Pokud zahrnete tento atribut, úvodní a koncové mezery se odstraní, zbývající řetězec se ořízne na 32 znaků a výsledek se předá do ukončení. Pokud tento atribut vynecháte, bude pro ukončení předána výchozí hodnota 32 mezer.

Maximální délka tohoto pole je 32 znaků.

- Identifikujte posloupnost tohoto ukončení ve vztahu k ostatním uživatelským procedurám pomocí následujícího atributu:

  Sequence=sequence_number
  Posloupnost, ve které je tato uživatelská procedura rozhraní API volána vzhledem k jiným uživatelským procedurám rozhraní API. Uživatelská procedura s nízkým pořadovým číslem se volá před ukončením s vyšším pořadovým číslem. Není třeba, aby pořadové číslování východů bylo souvislé. Posloupnost 1, 2, 3 má stejný výsledek jako posloupnost 7, 42, 1096. Pokud mají dvě uživatelské procedury stejné pořadové číslo, rozhodne správce front, který z nich má volat jako první. Můžete říci, které bylo voláno po události, tím, že jste čas nebo značku v oblasti ExitChainoznačené jako ExitChainAreaPtr v MQAXP nebo zápisem vašeho vlastního souboru protokolu.

  Tento atribut je nepodepsaná číselná hodnota.

Ukázkové stanzy

Ukázkový soubor mqs.ini obsahuje následující oddíly:

ApiExitTemplate
Tato stanza definuje ukončení s popisným názvem `OurPayrollQueueAuditor`, názvem modulu `auditora` pořadovým číslem 2. Datová hodnota 123 je předána uživatelské proceduře.

ApiExitCommon
Tato stanza definuje ukončení s popisným názvem `MQPoliceman`, názvem modulu `tmqpa` pořadovým číslem 1. Předané údaje jsou instrukce (`CheckEverything`).

```
mqs.ini

   ApiExitTemplate:
      Name=OurPayrollQueueAuditor
      Sequence=2
      Function=EntryPoint
      Module=/usr/ABC/auditor
      Data=123
   ApiExitCommon:
      Name=MQPoliceman
      Sequence=1
      Function=EntryPoint
      Module=/usr/MQPolice/tmqp
      Data=CheckEverything
```

Následující ukázkový soubor qm.ini obsahuje lokální definici ApiExitpro ukončení s popisným názvem `ClientApplicationAPIchecker`, názvem modulu `ClientAppCheckera` pořadovým číslem 3.

```
qm.ini

   ApiExitLocal:
      Name=ClientApplicationAPIchecker
      Sequence=3
      Function=EntryPoint
      Module=/usr/Dev/ClientAppChecker
      Data=9.20.176.20
```

Kanály-uživatelské programy pro kanály systému zpráv

Tato kolekce témat obsahuje informace o programech výstupních bodů kanálu WebSphere MQ pro kanály systému zpráv.

Agenti kanálu zpráv (MCA) mohou také volat uživatelské procedury pro převod dat. Další informace o zápisu uživatelských procedur pro převod dat, viz “Zápis uživatelských procedur pro převod dat” na stránce 399.

Některé z těchto informací platí také pro uživatelské procedury na kanálech MQI, které připojují klienty WebSphere MQ MQI ke správcům front. Další informace naleznete v tématu Programy ukončení kanálů pro kanály MQI.

Programy výstupních bodů kanálu jsou volány v definovaných místech zpracování prováděné programy MCA.

Některé z těchto programů uživatelských procedur pracují v komplementárních párech. Je-li například odesílající agent MCA volán odesílajícím programem MCA pro šifrování zpráv pro přenos, musí na konci procesu ukončit proces, který je komplementární, aby mohl být proces dekonstrukci ukončen.

Tabulka 55 na stránce 381 zobrazuje typy ukončení kanálu, které jsou dostupné pro každý typ kanálu.

Tabulka 55. Uživatelské procedury kanálu jsou dostupné pro každý typ kanálu.

Typ kanálu	Ukončení zprávy	Ukončení opakování zprávy	Ukončení příjmu	Uživatelská procedura pro zabezpečení zprávy	Ukončení odeslání	Uživatelská procedura automatické definice
Kanál odesílatele	Ano		Ano	Ano	Ano	
Kanál serveru	Ano		Ano	Ano	Ano	
Kanál odesílatele klastru	Ano		Ano	Ano	Ano	Ano
Kanál příjemce	Ano	Ano	Ano	Ano	Ano	Ano
Kanál žadatele	Ano	Ano	Ano	Ano	Ano	
Přijímací kanál klastru	Ano	Ano	Ano	Ano	Ano	Ano
Kanál připojení klienta			Ano	Ano	Ano	
Kanál připojení serveru			Ano	Ano	Ano	Ano

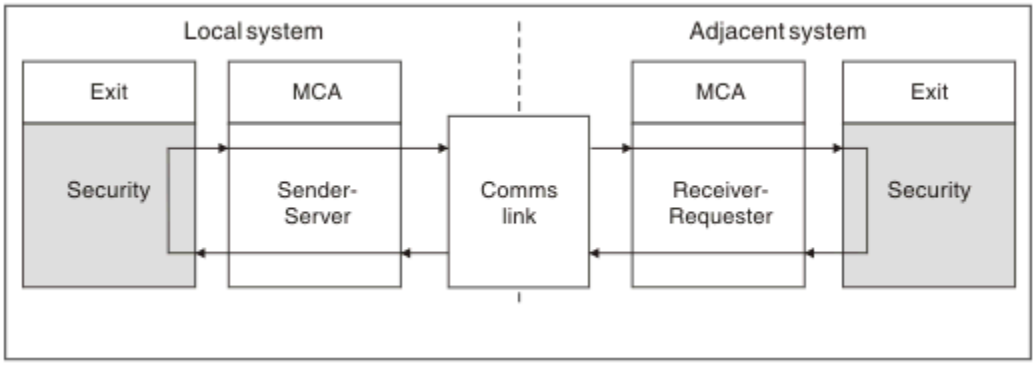
Chcete-li spustit uživatelské procedury kanálu na straně klienta, nelze použít proměnnou prostředí MQSERVER. Namísto toho vytvořte a vytvořte odkaz na tabulku definic kanálů klienta (CCDT), jak je popsáno v tématu Tabulka definic kanálů klienta .

Přehled zpracování

Přehled toho, jak MCAs používají programy uživatelské procedury kanálu.

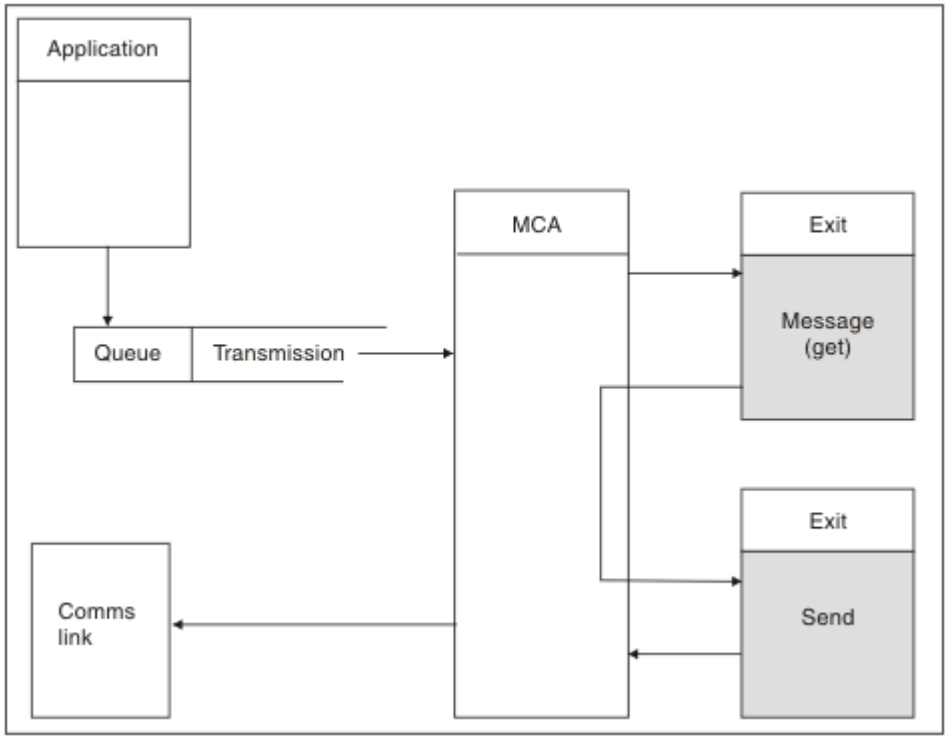
Při spuštění se při spuštění dialogového okna MMCA spustí dialogové okno spuštění synchronizace. Poté se přepnou na výměnu dat, která zahrnuje i uživatelské procedury zabezpečení. Tyto uživatelské procedury musí být úspěšně ukončeny pro dokončení fáze spuštění a aby bylo možné přenášet zprávy.

Fáze kontroly zabezpečení je smyčka, jak je zobrazeno v části Obrázek 77 na stránce 382.

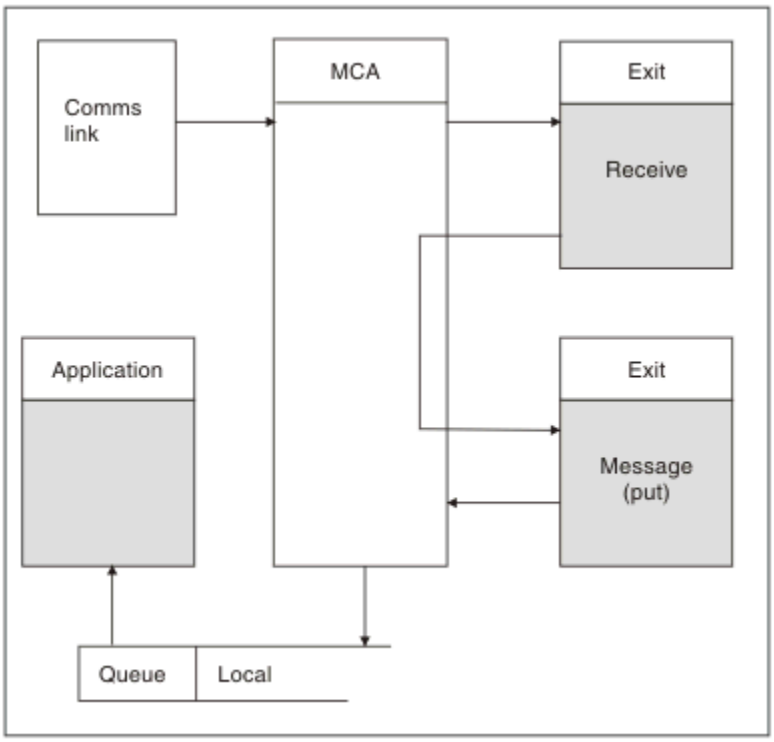


Obrázek 77. Smyčka ukončení zabezpečení

Během fáze přenosu zpráv odesílající agent MCA získává zprávy z přenosové fronty, volá uživatelskou proceduru zprávy, zavolá uživatelskou proceduru odeslání a odešle zprávu přijímajícímu programu MCA, jak ukazuje téma Obrázek 78 na stránce 382.



Obrázek 78. Příklad uživatelské procedury odeslání na konci kanálu zpráv odesílatele



Obrázek 79. Příklad uživatelské procedury příjmu na přijímacím konci kanálu zpráv

Přijímající agent MCA přijme zprávu z komunikačního spojení, zavolá uživatelskou proceduru pro přijetí zprávy, zavolá uživatelskou proceduru zprávy a pak umístí zprávu do lokální fronty, jak ukazuje Obrázek 79 na stránce 383. (Uživatelská procedura příjmu může být volána více než jednou, než se zavolá uživatelská procedura pro ukončení zprávy.)

Psaní programů výstupních bodů kanálu

Můžete použít následující informace, které vám pomohou psát programy výstupního bodu kanálu.

Uživatelské procedury a programy výstupního bodu kanálu mohou používat všechna volání MQI, s výjimkou těch, které jsou uvedeny v následujících oddílech. Pro produkt MQ V7 a novější obsahuje struktura MQCXP verze 7 a vyšší popisovač připojení hConn, který lze použít místo zadání volání MQCONN. Pro dřívější verze je nutné zadat příkaz MQCONN, i když je vráceno varování MQRC_ALREADY_CONNECTED, protože samotný kanál je připojen ke správci front.

Všimněte si, že uživatelská procedura kanálu musí zajišťovat neporušenost vláken.

U uživatelských procedur v kanálech připojení klienta závisí správce front, k němuž se pokus o připojení pokusí, závislá na tom, jak byla uživatelská procedura propojena. Pokud byla uživatelská procedura propojena s parametrem MQM.LIB a neurčíte název správce front v rámci volání MQCONN, pokusí se uživatelská procedura připojit k výchozímu správci front ve vašem systému. Pokud byla uživatelská procedura propojena s parametrem MQM.LIB a zadáte název správce front, který byl předán uživatelské proceduře prostřednictvím pole QMgrName objektu MQCD, se uživatelská procedura pokusí o připojení k tomuto správci front. Pokud byla uživatelská procedura propojena s MQIC.LIB nebo jakákoli jiná knihovna, volání MQCONN selže, ať již uvedete název správce front nebo ne.

Měli byste se vyhnout změně stavu transakce přidružené k předanému hConn ve výstupu kanálu; nesmíte použít příkazy MQCMIT, MQBACK nebo MQDISC s kanálem hConna nemůžete použít příkazové slovo MQBEGIN, které určuje kanál hConn.

Je-li MQCONNX použit při specifikaci MQCNO_HANDLE_SHARE_BLOCK nebo MQCNO_HANDLE_SHARE_NO_BLOCK k vytvoření nového připojení k produktu IBM WebSphere MQ , je vaší zodpovědností zajistit správnou správu připojení a odpojení od správce front. Například uživatelská

procedura kanálu, která vytváří nové připojení ke správci front při každém vyvolání bez odpojení, má za následek sestavení připojení a zvýšení počtu podprocesů agenta.

Ukončení se spouští ve stejném podprocesu jako samotný agent MCA a používá stejný popisovač připojení. Takže běží uvnitř stejné UOW jako MCA a všechna volání provedená v rámci synchronizačního bodu jsou potvrzována nebo vrácena kanálem na konci dávky.

Proto může uživatelská procedura kanálu pro zprávy odesílat zprávy s oznámením, které jsou potvrzeny pouze do této fronty, je-li potvrzena dávka obsahující původní zprávu. Je tedy možné vyslat volání MQI bodu synchronizace z uživatelské procedury pro zprávy kanálu.

Uživatelská procedura kanálu může měnit pole na disku MQCD. Tyto změny se však nepodnily, kromě případů uvedených v uvedených okolnostech. Pokud uživatelský program kanálu změní pole ve struktuře dat MQCD, bude nová hodnota pro proces kanálu IBM WebSphere MQ ignorována. Nová hodnota však zůstane na MQCD a je předávána všem zbývajícím uživatelským procedurám v řetězu ukončení a v libovolné konverzaci sdílející instanci kanálu. Další informace naleznete v tématu Změna polí MQCD v uživatelské proceduře kanálu

Také v případě programů napsaných v jazyku C nesmí být funkce knihovny non-reentrant jazyka C použita v programu ukončovacím programem kanálu.

Pokud použijete více knihoven ukončení kanálu současně, mohou nastat problémy na některých platformách produktu UNIX and Linux , pokud kód pro dva různé uživatelské procedury obsahuje stejně pojmenované funkce. Je-li načtena uživatelská procedura kanálu, dynamický zavaděč v knihovně uživatelských procedur interpretuje názvy funkcí na adresy, na nichž je knihovna načtena. Pokud dvě uživatelské knihovny definují samostatné funkce, které mají identické názvy, může tento proces rozpoznání nesprávně interpretovat názvy funkcí jedné knihovny, aby používaly funkce jiné. Pokud se vyskytne tento problém, uveďte linker, že musí exportovat pouze požadované funkce exit a MQStart, protože tyto funkce nejsou ovlivněny. Jiné funkce musí mít lokální viditelnost, aby se nepoužívaly pro funkce mimo vlastní knihovnu uživatelské procedury. Další informace naleznete v dokumentaci k řádku sestavovacího programu.

Všechny uživatelské procedury jsou volány spolu se strukturou parametrů uživatelské procedury kanálu (MQCXP), strukturou definice kanálu (MQCD), připravenou vyrovnávací pamětí dat, parametrem délky dat a parametrem délky vyrovnávací paměti. Délka vyrovnávací paměti nesmí být překročena:

- Pro ukončení zpráv je třeba povolit, aby největší zpráva byla odeslána přes kanál, a dále délka struktury MQXQH.
- Pro uživatelské procedury odeslání a přijetí je největší vyrovnávací paměť, kterou musíte povolit, takto:

  LU 6.2
  32 kB

  TCP:
  32 kB

  Poznámka: Maximální použitelná délka může být o 2 bajty menší než tato délka. Pro podrobnosti zkontrolujte hodnotu vrácenou v MaxSegmentLength. Další informace o délce trvání MaxSegmentnaleznete v části MaxSegmentLength.

  NetBIOS:
  64 KB

  SPX:
  64 KB

  Poznámka: Uživatelské procedury příjmu odesílacích kanálů a uživatelské procedury odesílatele v přijímacích kanálech používají pro protokol TCP vyrovnávací paměti 2 kB.
- Pro uživatelské procedury zabezpečení přiděluje prostředek distribuované fronty vyrovnávací paměť o velikosti 4000 bajtů.

Je přípustné, aby východ vrátil náhradní pufr, spolu s příslušnými parametry. Podrobnosti o volání naleznete v příručce “Kanály-uživatelské programy pro kanály systému zpráv” na stránce 380 .

Psaní ukončovacích programů kanálů v systémech Windows, UNIX and Linux
Následující informace vám pomohou při psaní programů ukončovacích kanálů pro systémy Windows, UNIX and Linux .

Postupujte podle pokynů uvedených v tématu “Uživatelské procedury pro psaní a kompilaci a instalovatelné služby” na stránce 358. Pokud je to vhodné, použijte následující informace specifické pro výstupní bod kanálu:

Uživatelská procedura musí být napsána v jazyce C a je to knihovna DLL v systému Windows.

Definujte v uživatelské proceduře fiktivní rutinu MQStart () a jako vstupní bod určete `MQStart` jako vstupní bod v knihovně. Obrázek 80 na stránce 385 ukazuje, jak nastavit položku pro váš program:

```
#include <cmqec.h>

void MQStart() {;} /* dummy entry point - for consistency only */
void MQENTRY ChannelExit ( PMQCXP  pChannelExitParms,
                           PMQCD   pChannelDefinition,
                           PMQLONG pDataLength,
                           PMQLONG pAgentBufferLength,
                           PMQVOID pAgentBuffer,
                           PMQLONG pExitBufferLength,
                           PMQPTR  pExitBufferAddr)
{
... Insert code here
}
```

Obrázek 80. Ukázkový zdrojový kód pro uživatelskou proceduru kanálu

Při zápisu kanálů pro systém Windows s použitím jazyka Visual C + + je třeba vytvořit vlastní soubor DEF . Příklad toho, jak je ukázáno v Obrázek 81 na stránce 385. Další informace o zápisu ukončovacích programů kanálu najdete v tématu “Psaní programů výstupních bodů kanálu” na stránce 383.

```
EXPORTS
ChannelExit
```

Obrázek 81. Ukázkový soubor DEF pro systém Windows

Ukončovací programy zabezpečení kanálu
Ukončovací programy zabezpečení můžete použít k ověření, že partner na druhém konci kanálu je pravý. To je známé jako ověření. Chcete-li určit, že kanál musí používat uživatelskou proceduru pro zabezpečení zprávy, zadejte do pole SCYEXIT definice kanálu název uživatelské procedury.

Poznámka: Ověření může být také dosaženo pomocí záznamů ověření kanálu. Záznamy ověření kanálu poskytují skvělou flexibilitu při prevenci přístupu ke správcům front z určitých uživatelů a kanálů a při mapování vzdálených uživatelů na identifikátory uživatelů produktu IBM WebSphere MQ . Podpora SSL a TLS je také poskytována produktem IBM WebSphere MQ k ověření vašich uživatelů a k zajištění šifrování a integrity dat pro vaše data. Další informace o zabezpečení SSL a TLS naleznete v příručce WebSphere MQ support for SSL and TLS. Pokud však stále vyžadujete propracovanější (nebo odlišné) formy zpracování zabezpečení a další typy kontrol a zabezpečení kontextu zabezpečení, zvažte použití uživatelských procedur pro zabezpečení zápisu.

Pro uživatelské procedury zabezpečení zapsané před IBM WebSphere MQ Version 7.1 stojí za zmínku, že dřívější verze produktu IBM WebSphere MQ dotazovali základního zabezpečeného soketu (např. GSKit) k určení rozlišujícího názvu partnera certifikátu vzdáleného partnera (SSLPEER) a rozlišovacího jména vydávajícího (SSLCERTI). V podpoře produktu IBM WebSphere MQ Version 7.1 byla přidána podpora pro rozsah nových atributů zabezpečení. Chcete-li získat přístup k těmto atributům, produkt IBM WebSphere MQ Version 7.1 získá kódování DER certifikátu a použije jej k určení DN subjektu a vydavatele. Atributy DN subjektu a vydavatele se zobrazují v následujících atributech stavu kanálu:

- SSLPEER (PCF selektor MQCACH_SSL_SHORT_PEER_NAME)
- SSLCERTI (PCF selector MQCACH_SSL_CERT_ISSUER_NAME)

Tyto hodnoty jsou vraceny příkazy pro stav kanálu a také data předaná uživatelským procedurám zabezpečení kanálu, jak je zobrazeno:

- MQCD SSLPeerNamePtr
- MQCXP SSLRemCertIssNamePtr

Atribut SERIALNUMBER v produktu IBM WebSphere MQ Version 7.1je také obsažen v DN subjektu a obsahuje sériové číslo pro certifikát vzdáleného partnera. Také některé atributy DN jsou vráceny v jiné posloupnosti z předchozích vydání. Následně se změní složení polí SSLPEER a SSLCERTI v produktu Version 7.1 z předchozích verzí, a proto se doporučuje, aby všechny bezpečnostní východy nebo aplikace závislé na těchto polích byly zkontrolovány a aktualizovány.

Existující filtry názvů rovnocenných uzlů WebSphere MQ určené prostřednictvím pole SSLPEER v definici kanálu nebudou ovlivněny a budou nadále fungovat stejným způsobem jako v předchozích verzích. Důvodem je skutečnost, že odpovídající algoritmus názvu partnera produktu WebSphere MQ byl aktualizován tak, aby zpracoval existující filtry SSLPEER, aniž by bylo nutné měnit definice kanálu. Tato změna s největší pravděpodobností ovlivní uživatelské procedury zabezpečení a aplikace, které závisejí na hodnotě DN subjektu a DN vydávajícího programového rozhraní PCF.

Uživatelská procedura zabezpečení může být napsána v jazyce C nebo Java.

Ukončovací programy zabezpečení kanálu jsou volány na následujících místech v cyklu zpracování agenta MCA:

- Při inicializaci a ukončení programu MCA.
- Okamžitě po dokončení počátečního vyjednávání dat při spuštění kanálu. Přijímač nebo konec serveru kanálu může iniciovat výměnu zpráv o zabezpečení se vzdáleným koncem tím, že poskytuje zprávu, která má být doručena do ukončení zabezpečení na vzdáleném konci. Může se také stát, že k tomu bude úpadek. Ukončovací program se spustí znovu, aby zpracoval jakoukoli zprávu o zabezpečení přijatou ze vzdáleného ukončení.
- Okamžitě po dokončení počátečního vyjednávání dat při spuštění kanálu. Odesilatel nebo žadatel o ukončení kanálu zpracovává zprávu zabezpečení přijatou ze vzdáleného ukončení, nebo iniciuje výměnu zabezpečení, když vzdálený konec nemůže. Ukončovací program se spustí znovu, aby zpracoval všechny následné zprávy zabezpečení, které mohou být přijaty.

Žadatelský kanál se nikdy nevolá s rozhraním MQXR_INIT_SEC. Kanál oznamuje serveru, že má uživatelský program zabezpečení, a server pak má možnost zahájit uživatelskou proceduru pro zabezpečení zprávy. Pokud ji nemá, informuje o tom žadatele a tok s nulovou délkou se vrátí do výstupního programu.

Poznámka: Vyhněte se odesílání zpráv zabezpečení s nulovou délkou.

Příklady dat vyměněných zabezpečovacími programy jsou ilustrovány v číslech Obrázek 82 na stránce 387 až Obrázek 85 na stránce 389. Tyto příklady ukazují pořadí událostí, které se vyskytnou při ukončení bezpečnostní procedury zásobníku, a ukončení zabezpečení odesílatele. Postupné řádky v číslech představují plynutí času. V některých případech se události na příjemce a odesilateli nekorelují, a proto se mohou vyskytnout ve stejnou dobu nebo v různých časech. V jiných případech událost na jednom ukončovacím programu má za následek doplňkovou událost, která se vyskytne později v jiném ukončovacím programu. Například v Obrázek 82 na stránce 387:

1. Příjemce a odesilatel jsou vyvolány s MQXR_INIT, ale tato vyvolání nejsou korelovaná a mohou se proto vyskytnout ve stejnou dobu nebo v různých časech.
2. Přijímač je dále vyvolán s MQXR_INIT_SEC, ale vrací MQXCC_OK, které nevyžaduje žádnou doplňkovou událost u uživatelské procedury odesílatele.
3. Odesilatel je nyní vyvolán s MQXR_INIT_SEC. Tato hodnota není korelována s vyvoláním přijímače s MQXR_INIT_SEC. Odesilatel vrátí zprávu MQXCC_SEND_SEC_MSG, která způsobí doplňkovou událost při ukončení přijímače.
4. Příjemce se pak vyvolá s MQXR_SEC_MSG a vrátí MQXCC_SEND_SEC_MSG, což způsobí doplňkovou událost u uživatelské procedury pro odeslání zprávy.
5. Odesilatel je pak vyvolán s MQXR_SEC_MSG a vrací MQXCC_OK, což nevyžaduje žádnou doplňkovou událost na ukončení přijímače.

Receiver exit	Sender exit
Invoked with MQXR_INIT Responds with MQXCC_OK	Invoked with MQXR_INIT Responds with MQXCC_OK
Invoked with MQXR_INIT_SEC Responds with MQXCC_OK	
	Invoked with MQXR_INIT_SEC Responds with MQXCC_SEND_SEC_MSG
Invoked with MQXR_SEC_MSG Responds with MQXCC_SEND_SEC_MSG	
	Invoked with MQXR_SEC_MSG Responds with MQXCC_OK
Message transfer begins	

Obrázek 82. Výměna iniciovaná odesilatelem se smlouvou

Receiver exit	Sender exit
Invoked with MQXR_INIT Responds with MQXCC_OK	Invoked with MQXR_INIT Responds with MQXCC_OK
Invoked with MQXR_INIT_SEC Responds with MQXCC_OK	
	Invoked with MQXR_INIT_SEC Responds with MQXCC_SEND_SEC_MSG
Invoked with MQXR_SEC_MSG Responds with MQXCC_OK	
	Invoked with MQXR_SEC_MSG Responds with MQXCC_SUPPRESS_FUNCTION
Channel closes	
Invoked with MQXR_TERM Responds with MQXCC_OK	Invoked with MQXR_TERM Responds with MQXCC_OK

Obrázek 83. Výměna iniciovaná odesilatelem bez shody

Receiver exit	Sender exit
Invoked with MQXR_INIT Responds with MQXCC_OK	Invoked with MQXR_INIT Responds with MQXCC_OK
Invoked with MQXR_INIT_SEC Responds with MQXCC_SEND_SEC_MSG	
	Invoked with MQXR_SEC_MSG Responds with MQXCC_SEND_SEC_MSG
Invoked with MQXR_SEC_MSG Responds with MQXCC_OK	
Message transfer begins	
Invoked with MQXR_TERM Responds with MQXCC_OK	Invoked with MQXR_TERM Responds with MQXCC_OK

Obrázek 84. Výměnu zahájená příjemcem s dohodou

Receiver exit	Sender exit
Invoked with MQXR_INIT Responds with MQXCC_OK	Invoked with MQXR_INIT Responds with MQXCC_OK
Invoked with MQXR_INIT_SEC Responds with MQXCC_SEND_SEC_MSG	
	Invoked with MQXR_SEC_MSG Responds with MQXCC_OK
Invoked with MQXR_SEC_MSG Responds with MQXCC_SUPPRESS_FUNCTION	
Channel closes	

Obrázek 85. Výměnu přijímanou přijímačem bez dohody

Uživatelský program zabezpečení kanálu předává vyrovnávací paměť agenta obsahující data zabezpečení, kromě všech záhlaví přenosu generovaných uživatelskou procedurou pro zabezpečení zprávy. Tato data mohou být vhodná k tomu, aby bylo možné provést ověření zabezpečení buď na konci kanálu.

Uživatelský program zabezpečení na odesílajícím a přijímajícím na konci kanálu zpráv může vrátit buď dva kódy odezvy pro jakékoli volání:

- Výměna zabezpečení skončila bez chyb
- Potlačit kanál a zavřít jej

Poznámka:

1. Uživatelské procedury zabezpečení kanálu obvykle pracují ve dvojicích. Definujete-li vhodné kanály, ujistěte se, že jsou pro oba konce kanálu pojmenovány kompatibilní uživatelské programy.
2. V produktu IBM imohou být programy ukončení zabezpečení, které byly kompilovány s oprávněním "Použít adoptované oprávnění" (USEADPAUT = *YES), adoptovat oprávnění QMQM nebo QMQMADM. Je třeba dbát na to, aby tato uživatelská procedura nepoužívala tuto funkci k vytvoření bezpečnostního rizika pro váš systém.
3. Na kanálu SSL, na kterém druhý konec kanálu poskytuje certifikát, obdrží uživatelská procedura zabezpečení rozlišující název předmětu tohoto certifikátu v poli MQCD, ke kterému má přístup SSLPeerNamePtr, a rozlišující název vydavatele v poli MQCXP, ke kterému má přístup SSLRemCertIssNamePtr. Používá se k tomu, který název může být umístěn:
   - Omezte přístup přes kanál SSL.
   - Chcete-li změnit MQCD.MCAUserIdentifier založený na názvu.

Související pojmy

Záznamy ověření kanálu

Koncepce zabezpečení SSL (Secure Sockets Layer) a TLS (Transport Layer Security)

Psaní uživatelské procedury zabezpečení

Uživatelská procedura zabezpečení můžete zapsat pomocí kódu kostry ukončení zabezpečení.

Obrázek 86 na stránce 390 ilustruje, jak napsat uživatelskou proceduru zabezpečení.

```
void MQENTRY MQStart() {;}
void MQENTRY EntryPoint (PMQVOID pChannelExitParms,
                         PMQVOID pChannelDefinition,
                         PMQLONG pDataLength,
                         PMQLONG pAgentBufferLength,
                         PMQVOID pAgentBuffer,
                         PMQLONG pExitBufferLength,
                         PMQPTR  pExitBufferAddr)
{
  PMQCXP pParms = (PMQCXP)pChannelExitParms;
  PMQCD  pChDef = (PMQCD)pChannelDefinition;
  /* TODO: Add Security Exit Code Here */

}
```

Obrázek 86. Kód kostry uživatelské procedury zabezpečení

Je třeba, aby existoval standardní vstupní bod MQStart produktu WebSphere MQ , ale není vyžadován k provedení žádné funkce. Název funkce (EntryPoint v tomto příkladu) může být změněn, ale funkce musí být exportována, když je knihovna kompilována a propojena. Stejně jako v předchozím příkladu musí být ukazatele pChannelExitParms přetypovány na PMQCXP a definice pChannelmusí být přetypovány na PMQCD. Všeobecné informace o ukončení volání kanálu a použití parametrů naleznete v tématu MQ_CHANNEL_EXIT. Tyto parametry se používají v proceduře zabezpečení následujícím způsobem:

PMQVOID pChannelExitParms
Vstup a výstup

Ukazatel na strukturu MQCXP-cast na PMQCXP pro přístup k polím. Tato struktura se používá ke komunikaci mezi uživatelskou procedurou a agentem MCA. Následující pole v aplikaci MQCXP mají zvláštní význam pro uživatelské procedury zabezpečení:

ExitReason
Sděluje zabezpečení Exit o aktuálním stavu v rámci výměny zabezpečení a používá se při rozhodování o tom, jaká akce se má provést.

ExitResponse
Odezva na agenta MCA, který diktuje další fázi v rámci výměny zabezpečení.

ExitResponse2
Přebytečné řídicí příznaky pro řízení způsobu, jakým program MCA interpretuje odezvu ukončení zabezpečení.

Oblast ExitUser
16 bajtů (maximum) úložiště, které může být použito uživatelskou procedurou zabezpečení k udržování stavu mezi voláními.

ExitData
Obsahuje data uvedená v poli SCYDATA definice kanálu (32 bajtů směrem doprava s mezerami).

Definice PMQVOID pChannel
Vstup a výstup

Ukazatel na strukturu MQCD-cast na PMQCD pro přístup k polím. Tento parametr obsahuje definici kanálu. Následující pole v produktu MQCD jsou zvláště zajímavá pro uživatelské procedury zabezpečení:

ChannelName
Název kanálu (20 bajtů směrem doprava s mezerami).

ChannelType
Kód definující typ kanálu.

Identifikátor uživatele MCA
Tato skupina tří polí je inicializována na hodnotu pole MCAUSER, která je uvedena v definici kanálu. Jakýkoli identifikátor uživatele uvedený v poli zabezpečení v těchto polích se používá pro řízení přístupu (nepoužitelné pro kanály SDR, SVR, CLNTCONN nebo CLUSSDR).

MCAUserIdentifier
Prvních 12 bajtů identifikátoru doplněného doprava o prázdné místo.

LongMCAUserIdPtr
Ukazatel na vyrovnávací paměť obsahující identifikátor plné délky (nezaručený nezajištěný null) má přednost před MCAUserIdentifier.

LongMCAUserIdLength
Délka řetězce, na kterou ukazuje LongMCAUserIdPtr -musí být nastaveno, je-li nastavena hodnota LongMCAUserIdPtr .

Identifikátor vzdáleného uživatele
Vztahuje se pouze na dvojice kanálů CLNTCONN/SVRCONN. Není-li definována žádná uživatelská procedura zabezpečení CLNTCONN, budou tato tři pole inicializována klientem MCA klienta, takže mohou obsahovat identifikátor uživatele z prostředí klienta, který může být použit procedurou zabezpečení SVRCONN pro ověření a při určování identifikátoru uživatele MCA. Je-li definována uživatelská procedura zabezpečení CLNTCONN, pak tato pole nejsou inicializována a lze ji nastavit pomocí uživatelské procedury zabezpečení CLNTCONN nebo lze zprávy zabezpečení použít k předání identifikátoru uživatele z klienta na server.

Identifikátor RemoteUser
Prvních 12 bajtů identifikátoru bylo na pravé straně doplněno mezerami.

LongRemoteUserIdPtr
Ukazatel na vyrovnávací paměť obsahující identifikátor plné délky (nezaručený nezajištěný null) má přednost před identifikátorem RemoteUserIdentifier.

LongRemoteUserIdDélka
Délka řetězce, na kterou ukazuje LongRemoteUserIdPtr-, musí být nastavena, je-li nastavena hodnota LongRemoteUserIdPtr.

Délka PMQLONG pData
Vstup a výstup

Ukazatel na MQLONG. Obsahuje délku jakékoli procedury zabezpečení obsažené v AgentBuffer při vyvolání procedury zabezpečení. Musí být nastaveno uživatelskou procedurou zabezpečení na délku jakékoli zprávy odesílané v AgentBuffer nebo ExitBuffer.

PMQLONG pAgentBufferLength
Vstup

Ukazatel na MQLONG. Délka dat obsažených v AgentBuffer při vyvolání uživatelské procedury zabezpečení.

Vyrovnávací paměť PMQVOID pAgent
Vstup a výstup

Při vyvolání uživatelské procedury zabezpečení tato zpráva odkazuje na jakoukoli zprávu odeslanou z uživatelské procedury pro ukončení práce. Má-li parametr ExitResponse2 ve struktuře MQCXP nastavený příznak MQXR2_USE_AGENT_BUFFER (výchozí), musí zabezpečení ukončit tento parametr tak, aby ukazoval na odesílaná data zpráv.

PMQLONG pExitBufferLength
Vstup a výstup

Ukazatel na MQLONG. Tento parametr je inicializován na 0 při prvním vyvolání uživatelské procedury zabezpečení a vrácená hodnota se udržuje mezi voláními zabezpečení během výměny zabezpečení.

PMQPTR pExitBufferAddr
Vstup a výstup

Tento parametr se inicializuje na ukazatel Null při prvním vyvolání procedury zabezpečení a vrácená hodnota se bude udržovat mezi voláními zabezpečení během výměny zabezpečení. Je-li příznak MQXR2_USE_EXIT_BUFFER nastaven ve struktuře ExitResponse2 ve struktuře MQCXP, musí bezpečnostní procedura nastavit tento parametr tak, aby ukazovala na odesílaná data zprávy.

Rozdíly v chování mezi uživatelskými procedurami zabezpečení definovanými v párech kanálů CLNTCONN/SVRCONN a dalších dvojicích kanálů
Uživatelské procedury zabezpečení mohou být definovány na všech typech kanálů. Chování uživatelských procedur zabezpečení definovaných ve dvojicích kanálů CLNTCONN/SVRCONN se však mírně liší od uživatelských procedur zabezpečení definovaných v jiných párech kanálu.

Ukončení zabezpečení na kanálu CLNTCONN může nastavit identifikátor vzdáleného uživatele v definici kanálu pro zpracování partnerským výstupem SVRCONN nebo pro autorizaci OAM, pokud není definována žádná uživatelská procedura zabezpečení SVRCONN a pole MCAUSER v SVRCONN není nastaveno.

Není-li definována žádná uživatelská procedura zabezpečení CLNTCONN, pak je identifikátor vzdáleného uživatele v definici kanálu nastaven na identifikátor uživatele z klientského prostředí (který může být prázdný) klientem MCA.

Výměna zabezpečení mezi ukončenými prvky zabezpečení definovaná na dvojici kanálu CLNTCONN a SVRCONN je úspěšně dokončena, když funkce zabezpečení SVRCONN vrátí hodnotu ExitResponse MQXCC_OK. Výměna zabezpečení mezi ostatními dvojicemi kanálů je úspěšně dokončena, když uživatelská procedura zabezpečení, která iniciovala výměnu, vrátila ExitResponse MQXCC_OK.

Avšak kód MQXCC_SEND_AND_REQUEST_SEC_MSG ExitResponse lze použít k vynucení pokračování zabezpečení: Pokud je ExitResponse MQXCC_SEND_REQUEST_REQUEST_SEC_MSG vrácena pomocí CLNTCONN nebo SVRCONN Security Exit, musí partner ukončit odpověď odesláním zprávy zabezpečení (ne MQXCC_OK nebo null response) nebo se kanál ukončí. V případě uživatelských procedur zabezpečení definovaných v jiných typech kanálů se položka ExitResponse MQXCC_OK vrátila jako odezva na objekt MQXCC_SEND_AND_REQUEST_SEC_MSG z objektu Security Exit pro pokračování v rámci výměny zabezpečení, jako kdyby byla vrácena odezva s hodnotou Null, a nikoli při ukončení kanálu.

Uživatelská procedura zabezpečení SSPI
Produkt WebSphere MQ for Windows poskytuje proceduru zabezpečení, která poskytuje ověření pro kanály produktu WebSphere MQ pomocí rozhraní API zabezpečení služeb zabezpečení (SSPI). SSPI poskytuje integrovaná bezpečnostní zařízení systému Windows.

Tato uživatelská procedura pro zabezpečení dat je určena pro klienta WebSphere MQ i pro server WebSphere MQ .

Balíky zabezpečení jsou načteny z adresáře security.dll nebo secur32.dll. Tyto knihovny DLL se dodávají spolu s operačním systémem.

Jednosměrné ověření je poskytnuto v systému Windowspomocí ověřovacích služeb NTLM. Dvoucestné ověření je k dispozici v systému Windows 2000 s použitím ověřovacích služeb Kerberos .

Uživatelský program zabezpečení je dodáván ve zdrojovém formátu a ve formátu objektu. Kód objektu můžete použít tak, jak je, nebo můžete použít zdrojový kód jako výchozí bod k vytvoření svých vlastních uživatelských programů. Další informace o použití objektu nebo zdrojového kódu uživatelské procedury zabezpečení SSPI viz “Použití uživatelské procedury zabezpečení SSPI v systémech Windows” na stránce 163 .

Výstupní programy pro odesílání a příjem kanálů
Pomocí uživatelských procedur pro odesílání a příjem můžete provádět úlohy, jako je komprese dat a dekomprimace. Můžete určit seznam programů uživatelských procedur pro odeslání a přijetí, které mají být spuštěny v posloupnosti.

Ukončovací programy pro odesílání a příjem kanálů jsou volány na následujících místech v cyklu zpracování agenta MCA:

- Ukončovací programy pro odesílání a příjem jsou volány pro inicializaci programu MCA a pro ukončení programu MCA.
- Uživatelský program odeslání je vyvolán na jednom nebo na druhém konci kanálu, v závislosti na tom, kdy je odeslán přenos pro jeden přenos zprávy, okamžitě před odesláním přenosu přes linku. Poznámka 4 vysvětluje, proč jsou uživatelské procedury k dispozici v obou směrech i v případě, že kanály zpráv odesílají zprávy pouze v jednom směru.
- Uživatelský program příjmu je vyvolán na jednom nebo na druhém konci kanálu, v závislosti na tom, kdy byl přijat přenos pro jeden přenos zpráv, okamžitě po přenosu z odkazu. Poznámka 4 vysvětluje, proč jsou uživatelské procedury k dispozici v obou směrech i v případě, že kanály zpráv odesílají zprávy pouze v jednom směru.

Pro jeden přenos zpráv může existovat mnoho přenosů a může existovat mnoho iterací pro odesílací a přijímací výstupní programy před tím, než se zpráva dostane na konec zprávy na přijímajícím konci.

Ukončovací programy pro odeslání a přijetí kanálu jsou předávány vyrovnávací paměti agenta obsahující data přenosu, která jsou odeslána nebo přijata z komunikačního spojení. V případě ukončovacích programů pro odeslání je prvních 8 bajtů vyrovnávací paměti vyhrazeno pro použití agentem MCA a nesmí být změněno. Pokud program vrátí jinou vyrovnávací paměť, pak tyto první 8 bajtů musí existovat v nové vyrovnávací paměti. Formát dat prezentovaných pro ukončovací programy není definován.

Dobrý kód odezvy musí být vrácen ukončovacími programy pro odeslání a přijetí. Jakákoli jiná odpověď způsobí nestandardní ukončení agenta MCA (nestandardní konec).

Poznámka: Nevystavujte volání MQGET, MQPUT nebo MQPUT1 v rámci synchronizačního bodu z uživatelské procedury odeslání nebo přijetí.

Poznámka:

1. Uživatelské procedury pro odesílání a příjem obvykle pracují ve dvojicích. Například ukončení odeslání může komprimovat data a ukončení příjmu je dekomprimován nebo uživatelská procedura pro odeslání může zašifrovat data a dešifrovat ji při ukončení příjmu. Definujete-li vhodné kanály, ujistěte se, že jsou pro oba konce kanálu pojmenovány kompatibilní uživatelské programy.
2. Je-li pro kanál povolena komprese, jsou tyto uživatelské procedury předávány komprimovanými daty.

3. Uživatelské procedury odeslání a příjmu kanálu mohou být volány pro segmenty zpráv jiné než pro data aplikací, například stavové zprávy. Nejsou volány během dialogového okna spuštění ani fáze kontroly zabezpečení.
4. Ačkoli kanály zpráv odesílají zprávy pouze v jednom směru, data kanálového řízení, jako je například prezenční signál a ukončení dávkového zpracování, procházejí oběma směry a tyto východy jsou k dispozici v obou směrech také. Avšak některé z počátečních datových toků pro spuštění kanálu jsou vyloučeny ze zpracování některou z uživatelských procedur.
5. Existují okolnosti, za kterých mohou být uživatelské procedury odeslání a přijetí vyvolány mimo pořadí, například pokud spouštíte řadu ukončovacích programů nebo pokud spouštíte také ukončení zabezpečení. Poté, když je procedura příjmu poprvé volána pro zpracování dat, může přijmout data, která neprošla přes odpovídající uživatelskou proceduru odeslání. Pokud uživatelská procedura pro příjem právě provedla operaci, například dekomprese, aniž byste nejprve zkontrolovali, zda je tato operace vyžadována, výsledky by byly neočekávané.

   Musíte kódovat své uživatelské procedury pro odesílání a příjem takovým způsobem, že procedura příjmu může zkontrolovat, zda data, která přijímá, byla zpracována pomocí příslušné uživatelské procedury odeslání. Doporučeným způsobem, jak to provést, je kódovat uživatelské programy tak, aby:

   - Uživatelská procedura pro odeslání nastaví hodnotu devátého bajtu dat na 0 a před provedením operace posune všechna data po 1 bajtu. (Prvních 8 bajtů je vyhrazeno pro použití agentem MCA.)
   - Pokud uživatelská procedura příjmu přijme data s hodnotou 0 v bajtu 9, bude vědět, že data pocházejí z uživatelské procedury odeslání. Odstraní 0, provádí komplementární operaci a přesouvá výsledná data zpět o 1 bajt.
   - Pokud uživatelská procedura příjmu přijme data, která mají něco jiného než 0 v bajtu 9, předpokládá, že uživatelská procedura odeslání nebyla spuštěna, a odešle data zpět volajícímu beze změny.

   Při použití uživatelských procedur zabezpečení je kanál ukončen uživatelskou procedurou zabezpečení a lze ji volat bez příslušné uživatelské procedury pro přijetí zprávy. Jednou z možností, jak tomuto problému předejít, je kódovat uživatelskou proceduru zabezpečení nastavením příznaku v souboru MQCD.SecurityUserData nebo MQCD.SendUserData, například, když se uživatelská procedura rozhodne ukončit kanál. Poté musí uživatelská procedura odeslání kontrolovat toto pole a zpracovat data pouze v případě, že příznak není nastaven. Tato kontrola zabrání zbytečnému pozměnění dat odesláním ukončení odeslání, a tím zabrání výskytu chyb konverze, které by mohly nastat v případě, že uživatelská procedura zabezpečení přijala změněná data.

Programy ukončení odeslání kanálu-rezervace prostoru
Pomocí uživatelských procedur pro odesílání a přijímání můžete transformovat data před přenosem. Uživatelské programy odeslání kanálu mohou přidávat vlastní data o transformaci vyhrazením prostoru v přenosové vyrovnávací paměti.

Tato data jsou zpracována ukončovacím programem příjmu a pak jsou odebrána z vyrovnávací paměti. Můžete například chtít šifrovat data a přidat klíč zabezpečení pro dešifrování.

Jak rezervovat prostor a použít jej

Je-li volaný uživatelský program volán k inicializaci, nastavte pole *ExitSpace* MQXCP na počet bajtů, které mají být rezervovány. Podrobnosti viz MQCXP . *ExitSpace* lze nastavit pouze během inicializace, tj. pokud má *ExitReason* hodnotu MQXR_INIT. Je-li uživatelská procedura pro odeslání vyvolána bezprostředně před přenosem a *ExitReason* je nastavena na hodnotu MQXR_XMIT, jsou bajty *ExitSpace* vyhrazeny v přenosové vyrovnávací paměti. Produkt *ExitSpace* není podporován v systému z/OS.

Uživatelská procedura odeslání nemusí používat veškerý rezervovaný prostor. Může použít méně než *ExitSpace* bajtů nebo, pokud není vyrovnávací paměť pro přenos zaplněna, může uživatelská procedura použít více než vyhrazenou částku. Při nastavení hodnoty *ExitSpace*je nutné ponechat alespoň 1 kB dat zprávy v přenosové vyrovnávací paměti. Výkon kanálu může být ovlivněn, je-li vyhrazen vyhrazený prostor pro velké množství dat.

Co se děje na přijímajícím konci kanálu

Programy ukončení příjmu kanálu musí být nastaveny tak, aby byly kompatibilní s odpovídajícími uživatelskými procedurami odeslání. Uživatelské procedury příjmu musí znát počet bajtů ve vyhrazeném prostoru a musí v tomto prostoru odebrat data.

Hromadné ukončení odeslání

Můžete určit seznam programů uživatelských procedur pro odeslání a přijetí, které mají být spuštěny v posloupnosti. WebSphere MQ udržuje celkem prostor vyhrazený pro všechny uživatelské procedury odeslání. Tento celkový prostor musí ponechat alespoň 1 kB dat zprávy v přenosové vyrovnávací paměti.

Následující příklad ukazuje, jak je prostor přidělen pro tři uživatelské procedury odeslání, které byly volány za sebou:

1. Při volání pro inicializaci:
   - Odesílatel výstupu A rezervuje 1 KB.
   - Uživatelská procedura ukončení B rezervuje 2 kB.
   - Uživatelská procedura odeslání C rezervuje 3 kB.
2. Maximální velikost přenosu je 32 kB a uživatelská data jsou 5 kB dlouhá.
3. Ukončení A je voláno s 5 kB dat; je k dispozici až 27 kB, protože 5 KB je vyhrazeno pro uživatelské procedury B a C. Výstup A přidá 1 KB, množství, které je rezervované.
4. Ukončení B je voláno s 6 kB dat; je k dispozici až 29 kB, protože 3 KB je vyhrazeno pro ukončení C. Výstup B přidá 1 KB, méně než 2 kB, které je rezervované.
5. Výstup C je volán s 7 kB dat. K dispozici je až 32 kB. Exit C přidá 10K, více než 3 KB rezervované. Tato částka je platná, protože celkové množství dat, 17 KB, je menší než maximum 32 kB.

Ukončovací programy zpráv kanálu
Můžete použít uživatelskou proceduru zprávy kanálu k provedení úloh, jako je šifrování na odkazu, ověření nebo nahrazení příchozích ID uživatelů, konverze dat zprávy, žurnálování a zpracování referenčních zpráv. Můžete zadat seznam ukončovacích programů pro zprávy, které mají být spuštěny v posloupnosti.

Ukončovací programy pro zprávy kanálu jsou volány na následujících místech v cyklu zpracování agenta MCA:

- Při inicializaci a ukončení MCA
- Okamžitě poté, co odesílající agent MCA vydal volání MQGET
- Před přijetím příkazu MCA pro příjem volání MQPUT

Uživatelská procedura pro zprávy je předána do vyrovnávací paměti agenta obsahující záhlaví přenosové fronty, MQXQH a text zprávy aplikace načtený z fronty. (Formát MQXQH je uveden v MQXQH.) Pokud používáte referenční zprávy; tj. zprávy, které obsahují pouze záhlaví odkazující na nějaký jiný objekt, který má být odeslán, výstupní bod zprávy rozpozná záhlaví, MQRMH. Identifikuje objekt, načte jej jakýmkoli způsobem a připojí jej k záhlaví a předá jej do kanálu MCA pro přenos do přijímajícího agenta MCA. Na přijímajícím agentovi MCA rozpozná další uživatelská procedura zprávy, že tato zpráva je referenční zprávou, extrahuje objekt a předá záhlaví do cílové fronty. Chcete-li získat další informace o referenčních zprávách a některých ukázkových uživatelských procedurách, které je obsluhují, prohlédněte si téma “Referenční zprávy” na stránce 254 a “Spuštění ukázek referenční zprávy” na stránce 136 .

Uživatelské procedury pro zprávy mohou vrátit následující odpovědi:

- Odešlete zprávu (příkaz GET exit). Je možné, že zpráva byla změněna uživatelskou procedurou. (Tato funkce vrací MQXCC_OK.)
- Vložte zprávu do fronty (PUT exit). Je možné, že zpráva byla změněna uživatelskou procedurou. (Tato funkce vrací MQXCC_OK.)

- Nezpracujte tuto zprávu. Zpráva se umístí do fronty nedoručených zpráv (nedoručená fronta zpráv) pomocí agenta MCA.
- Zavřete kanál.
- Chybný návratový kód, který způsobí, že agent MCA bude abnormálně ukončen.

Poznámka:

1. Uživatelské procedury pro zprávy jsou volány jednou pro každou převedenou úplnou zprávu i v případě, že je zpráva rozdělena na části.
2. Pokud v systému UNIX poskytnete ukončení zprávy z jakéhokoli důvodu, nebude automatická konverze ID uživatelů na malá písmena fungovat. Viz Zabezpečení objektů v systémech UNIX and Linux.
3. Ukončení se spouští ve stejném podprocesu jako samotná MCA. Běží také uvnitř stejné pracovní jednotky (UOW) jako agent MCA, protože používá stejný popisovač připojení. Proto jsou všechny hovory provedené v rámci synchronizačního bodu potvrzeny nebo vráceny kanálem na konci dávky. Například jeden výstupní program zprávy kanálu může odesílat zprávy oznámení jinému a tyto zprávy jsou potvrzeny do fronty pouze tehdy, je-li potvrzena dávka obsahující původní zprávu.

   Proto je možné volat volání MQI rozhraní MQI z ukončovacího programu zpráv kanálu.

Převod zpráv mimo uživatelskou proceduru zprávy
Před voláním ukončení zprávy přijímá přijímající agent MCA některé převody na zprávu. Toto téma popisuje algoritmy používané k provádění převodů.

Která záhlaví se zpracovávají

Převodní rutina se spustí v MCA přijímače před voláním uživatelské procedury pro zpracování zprávy. Rutina konverze začíná hlavičkou MQXQH na začátku zprávy. Převodní rutina poté zpracuje zřetězená záhlaví, která následují za MQXQH, a v případě potřeby provádí konverzi. Zřetězená záhlaví mohou přesahovat odsazení obsažené v parametru HeaderLength dat MQCXP, která se předává do ukončení zprávy příjemce. Následující záhlaví jsou převedena na místo:

- MQXQH (název formátu "MQXMIT ")
- MQMD (toto záhlaví je součástí MQXQH a nemá žádný název formátu)
- MQMDE (název formátu "MQHMDE ")
- MQDH (název formátu "MQHDIST ")
- MQWIH (název formátu "MQHWIH ")

Následující záhlaví nejsou převedena, ale přešlápla na to, jak sběrnice MCA pokračuje v zpracování zřetězených záhlaví:

- MQDLH (název formátu "MQDEAD ")
- libovolná záhlaví s názvy formátů začínajících třemi znaky 'MQH' (například "MQHRF ") které nejsou jinak zmíněny

Způsob zpracování záhlaví

Parametr Formát každého záhlaví produktu WebSphere MQ je přečten agentem MCA. Parametr Formát je 8 bajtů v záhlaví, což je 8 jednobajtových znaků obsahujících název.

Agent MCA poté interpretuje data podle jednotlivých záhlaví jako typ pojmenovaného typu. Pokud se jedná o název typu záhlaví, který je vhodný pro převod dat produktu WebSphere MQ , bude převeden. Pokud se jedná o jiný název označující data mimo produktMQ (například MQFMT_NONE nebo MQFMT_STRING), pak program MCA zastaví zpracování záhlaví.

Co je MQCXP HeaderLength?

Parametr HeaderLength v datech MQCXP dodávaném do uživatelské procedury pro zprávy je celková délka záhlaví MQXQH (která zahrnuje záhlaví MQMD), MQMDE a MQDH na začátku zprávy. Tato záhlaví jsou zřetězená s použitím názvů a délek 'Formát'.

MQWIHKM

Zřetězená záhlaví mohou přesahovat mimo HeaderLength do oblasti uživatelských dat. Záhlaví MQWIH, je-li přítomno, je jedno z následujících záhlaví, které se zobrazují za záhlaví HeaderLength.

Je-li záhlaví MQWIH v řetězcích záhlavích, je před voláním ukončení zprávy přijímače převedeno na místo.

Ukončovací program opakování zprávy kanálu
Uživatelská procedura pro opakování zprávy kanálu je volána, když pokus o otevření cílové fronty nebude úspěšný. Pomocí uživatelské procedury můžete určit, za jakých okolností se má opakovat pokus, kolikrát se má opakovat a jak často.

Tato uživatelská procedura je také volána na přijímajícím konci kanálu v rámci inicializace a ukončení MCA.

Uživatelská procedura pro opakování zpráv kanálu je předána vyrovnávací paměti agenta obsahující záhlaví přenosové fronty, MQXQH a text zprávy aplikace načtený z fronty. Formát MQXQH je uveden v části Přehled pro MQXQH.

Ukončení je vyvoláno pro všechny kódy příčiny; uživatelská procedura určí, pro jaké kódy příčiny chce agent MCA opakovat, pro kolikrát a v jakých intervalech. (Hodnota sady počtu opakování zprávy při definování kanálu je předána této uživatelské proceduře na disku MQCD, ale tato hodnota může tuto hodnotu ignorovat.)

Pole Počet MsgRetryv MQCXP se zvyšuje o jedničku při vyvolání procedury MCA při každém vyvolání uživatelské procedury a uživatelská procedura vrací buď hodnotu MQXCC_OK s dobou čekání obsaženou v poli Interval MsgRetryMQCXP nebo MQXCC_SUPPRESS_FUNCTION. Opakované pokusy pokračují donekonečna, dokud uživatelská procedura nevrátí funkci MQXCC_SUPPRESS_FUNCTION v poli ExitResponse MQCXP. Informace o akcích provedených agentem MCA pro tyto kódy dokončení naleznete v dokumentu MQCXP .

Pokud budou všechna opakování neúspěšná, bude zpráva zapsána do fronty nedoručených zpráv. Není-li k dispozici žádná fronta nedoručených zpráv, kanál se zastaví.

Pokud nedefinujete uživatelskou proceduru opakování zpráv pro kanál a dojde k selhání, které pravděpodobně bude dočasné, například MQRC_Q_FULL, agent MCA použije počet opakování zpráv a intervaly opakování zpráv nastavené při definování kanálu. Je-li selhání trvalejší povahy a nedefinujete-li uživatelský program, zpráva se zapíše do fronty nedoručených zpráv.

Ukončovací program automatické definice kanálu
Uživatelská procedura automatické definice kanálu může být použita, když je přijat požadavek na spuštění přijímacího kanálu nebo kanálu připojení serveru, ale neexistuje žádná definice pro tento kanál (ne pro produkt WebSphere MQ pro z/OS). Lze ji také volat na všech platformách pro kanály odesílatele klastru a příjemce klastru, aby bylo možné upravit úpravu definice pro instanci kanálu.

Uživatelská procedura automatické definice kanálu může být volána na všech platformách s výjimkou operačního systému z/OS při přijetí požadavku na spuštění přijímacího kanálu nebo kanálu připojení serveru, ale neexistuje žádná definice kanálu. Můžete ji použít k úpravě zadané výchozí definice pro automaticky definovaného příjemce připojení nebo kanálu připojení serveru, SYSTEM.AUTO.RECEIVERnebo SYSTEM.AUTO.SVRCON. Popis, jak lze definice kanálů vytvořit automaticky, najdete v tématu Příprava kanálů .

Uživatelská procedura automatické definice kanálu může být volána také při přijetí požadavku ke spuštění kanálu odesílatele klastru. Lze ji volat pro kanály odesílatele klastru a příjemce klastru, aby bylo možné upravit úpravu definice pro tuto instanci kanálu. V takovém případě bude uživatelská procedura použita také pro produkt WebSphere MQ pro systém z/OS. Běžným použitím uživatelské procedury automatické definice kanálu je změna názvů ukončení platnosti zpráv (MSGEXIT, RCVEXIT, SCYEXIT a SENDEXIT), protože názvy uživatelských procedur mají různé formáty na různých platformách. Není-li zadána

žádná uživatelská procedura automatické definice kanálu, bude při výchozím chování v systému z/OS přezkoumávat název distribuované uživatelské procedury ve tvaru *[path]/libraryname(function)* a může obsahovat až osm znaků funkce, jsou-li přítomny, nebo názvy knihovny. V systému z/OSmusí výstupní program automatické definice kanálu změnit pole adresovaná prostřednictvím MsgExitPtr, MsgUserDataPtr, SendExitPtr, SendUserDataPtr, ReceiveExitPtr a ReceiveUserDataPtr, nikoli jako MsgExit, MsgUserData, SendExit, Datová pole SendUserData, ReceiveExit a ReceiveUserData.

Další informace najdete v tématu Automatická definice kanálů.

Stejně jako u jiných uživatelských procedur kanálů je seznam parametrů:

```
MQ_CHANNEL_AUTO_DEF_EXIT (ChannelExitParms, ChannelDefinition)
```

`ChannelExitParms` jsou popsány v MQCXP. `ChannelDefinition` je popsán v MQCD.

Objekt MQCD obsahuje hodnoty, které jsou použity ve výchozí definici kanálu, pokud tyto hodnoty nebyly při ukončení změněny. Uživatelská procedura může upravovat pouze část těchto polí; viz MQ_CHANNEL_AUTO_DEFEXIT. Pokus o změnu jiných polí však nezpůsobí chybu.

Uživatelská procedura automatické definice kanálu vrací odpověď MQXCC_OK nebo MQXCC_SUPPRESS_FUNCTION. Není-li vrácena žádná z těchto odpovědí, program MCA pokračuje ve zpracování, jako by byl vrácen objekt MQXCC_SUPPRESS_FUNKCE. To znamená, že automatická definice je opuštěna, není vytvořena žádná nová definice kanálu a kanál nelze spustit.

Kompilace uživatelských programů kanálů v systémech Windows, UNIX and Linux

Následující příklady vám pomohou při kompilaci programů výstupních bodů kanálů pro systémy Windows, UNIX and Linux .

Windows

Windows

Příkaz kompilátoru a propojovacího programu pro programy výstupního bodu 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 UNIX a Linux

Linux UNIX

V těchto příkladech je `exit` název knihovny a `ChannelExit` je název funkce. V systému AIX se soubor exportu nazývá `exit.exp`. Tato jména se používají v definici kanálu k odkazu na výstupní program s použitím formátu popsaného v tématu Definice kanálu MQCD-kanálu. Viz také parametr MSGEXIT příkazu DEFINE CHANNEL .

Ukázkové příkazy kompilátoru a propojovacího programu pro 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
```

Ukázkové příkazy kompilátoru a propojovacího programu pro uživatelské procedury kanálu v systému HP-UX

```
$ c89 +DD64 +z -c -D_HPUX_SOURCE -o exit.o exit.c -I/opt/mqm/inc
$ ld -b exit.o +ee MQStart +ee ChannelExit -o
/var/mqm/exits64/exit -L/usr/lib/pa20_64 -lpthread
$ rm exit.o
```

Ukázkové příkazy kompilátoru a propojovacího programu pro kanál-uživatelské procedury na platformách Linux , kde je správce front 32 bitů:

```
$ gcc -shared -fPIC -o /var/mqm/exits/exit exit.c -I/opt/mqm/inc
```

Ukázkové příkazy kompilátoru a propojovacího programu pro kanál-uživatelské procedury na platformách Linux , kde je správce front 64bitový:

```
$ gcc -m64 -shared -fPIC -o /var/mqm/exits64/exit exit.c -I/opt/mqm/inc
```

Ukázkové příkazy kompilátoru a propojovacího programu pro uživatelské procedury kanálu v systému Solaris:

```
$ cc -xarch=v9 -mt -G -o /var/mqm/exits64/exit exit.c -I/opt/mqm/inc
-R/usr/lib/64 -lsocket -lnsl -ldl
```

Na straně klienta lze použít 32bitovou nebo 64bitovou uživatelskou proceduru. Tato uživatelská procedura musí být propojena s parametrem mqic_r.

V systému AIXmusí být exportovány všechny funkce, které jsou volány produktem IBM WebSphere MQ . Ukázkový soubor exportu pro tento soubor make:

```
#!
channelExit
MQStart
```

Konfigurace uživatelských procedur kanálu

Chcete-li volat uživatelskou proceduru kanálu, je třeba ji pojmenovat v definici kanálu.

Uživatelské procedury kanálu musí být pojmenovány v definici kanálu. Toto pojmenování můžete provést, když poprvé definujete kanály, nebo můžete přidat informace později pomocí příkazu MQSC ALTER CHANNEL. Můžete také poskytnout názvy uživatelských procedur kanálu v datové struktuře kanálu MQCD. Formát názvu uživatelské procedury závisí na použité platformě IBM WebSphere MQ . Informace naleznete v dokumentu MQCD nebo Script (MQSC) Commands).

Pokud definice kanálu neobsahuje jméno programu uživatelské procedury, uživatelská procedura se nezavolá.

Uživatelská procedura automatické definice kanálu je vlastnost správce front, nikoli jednotlivým kanálem. Má-li být tato uživatelská procedura volána, musí být pojmenována v definici správce front. Chcete-li změnit definici správce front, použijte příkaz MQSC ALTER QMGR.

Zápis uživatelských procedur pro převod dat

Tato kolekce témat obsahuje informace o tom, jak psát uživatelské procedury pro zápis dat.

Poznámka: Nepodporováno v produktu MQSeries for VSE/ESA.

Provedete-li příkaz MQPUT, vytvoří aplikace deskriptor zprávy (MQMD) zprávy. Vzhledem k tomu, že produkt WebSphere MQ musí být schopen porozumět obsahu MQMD bez ohledu na platformu, na které je vytvořen, je systémem automaticky převeden.

Data aplikace však nejsou převedena automaticky. Pokud dochází k výměně znakových dat mezi platformami, kde se pole *CodedCharSetId* a *Encoding* liší, například mezi ASCII a EBCDIC, aplikace musí zajistit převod zprávy. Převod dat aplikací může provádět samotný správce front nebo uživatelský ukončovací program, který je označován jako *uživatelská procedura pro převod dat*. Správce front může provést vlastní převod dat pomocí jedné z vestavěných rutin pro převod, pokud se data aplikace nacházejí v jednom z vestavěných formátů (například MQFMT_STRING). Toto téma obsahuje informace o uživatelské proceduře pro převod dat, kterou produkt WebSphere MQ poskytuje, když data aplikace nejsou ve vestavěném formátu.

Řízení může být předáno do uživatelské procedury pro převod dat během volání MQGET. Tím se vyhnete převodu mezi různými platformami dříve, než dosáhnete konečného cíle. Je-li však konečným cílem platforma, která nepodporuje převod dat na MQGET, musíte zadat CONVERT (YES) na odesílacím kanálu, který odesílá data do svého konečného cíle. Tím je zajištěno, že produkt WebSphere MQ převede data během přenosu. V takovém případě musí být váš výstup pro převod dat umístěn v systému, ve kterém je definován odesílací kanál.

Volání MQGET je vydáno přímo aplikací. Nastavte pole *CodedCharSetId* a *Encoding* v deskriptoru MQMD na požadovanou znakovou sadu a kódování. Pokud vaše aplikace používá stejnou znakovou sadu a kódování jako správce front, nastavte parametr *CodedCharSetId* na hodnotu MQCCSI_Q_MGR a *Encoding* na hodnotu MQENC_NATIVE. Po dokončení volání MQGET mají tato pole odpovídající hodnoty pro vrácená data zprávy. Tyto hodnoty se mohou lišit od hodnot požadovaných v případě, že převod nebyl úspěšný. Aplikace by měla tato pole resetovat na hodnoty vyžadované před každým voláním MQGET.

Podmínky vyžadované pro uživatelskou proceduru pro převod dat, které mají být volány, jsou definovány pro volání MQGET v MQGET.

Popis parametrů předaných ukončovacím programu pro převod dat a podrobných poznámek k použití naleznete v tématu Převod dat pro volání MQ_DATA_CONV_EXIT a strukturu MQDXP.

Programy, které převádějí data aplikací mezi různými kódováními a identifikátory CCSID, musí odpovídat rozhraní pro převod dat produktu WebSphere MQ (DCI).

V případě zavedení klientů výběrového vysílání je možné na straně klienta spustit uživatelské procedury rozhraní API a uživatelské procedury pro převod dat, protože některé zprávy nemusí procházet správcem front. Následující knihovny jsou nyní součástí balíků klienta stejně jako balíky serveru:

Tabulka 56. Knihovny, které jsou nyní v balících klienta a serveru

Operační systém	**Knihovny**
Windows	32 bit & 64 bit: mqm.dll & mqm.pdb
Linux & HP-UX	32 bit & 64 bit: libmqm.so & libmqm_r.so
AIX	32 bit & 64 bit: libmqm.a & libmqm_r.a
Solaris	32 bitů & 64 bitů: libmqm.so

Vyvolání uživatelské procedury pro převod dat

Uživatelská procedura pro převod dat je uživatelská procedura, která přijímá řízení během zpracování volání MQGET.

Ukončení je vyvoláno, pokud jsou splněny následující podmínky:

- Volba MQGMO_CONVERT je určena v rámci volání MQGET.
- Některé nebo všechny z dat zprávy nejsou v požadované znakové sadě nebo kódování.
- Pole *Format* ve struktuře MQMD přidružené ke zprávě není MQFMT_NONE.
- Hodnota *BufferLength* zadaná v rámci volání MQGET není nulová.
- Délka dat zprávy není nula.
- Zpráva obsahuje data, která mají uživatelsky definovaný formát. Uživatelem definovaný formát může obsadit celou zprávu nebo může být předcházen jedním nebo více vestavěnými formáty. Například uživatelem definovaný formát může být před formátem MQFMT_DEAD_LETTER_HEADER. Ukončení je vyvoláno pro převod pouze uživatelem definovaného formátu; správce front převede všechny vestavěné formáty, které jsou před uživatelem definovaným formátem.

  Uživatelská procedura může být také vyvolána pro převod vestavěného formátu, ale k tomu dojde pouze v případě, že vestavěné převodní rutiny nemohou úspěšně převést vestavěný formát.

V poznámkách k použití volání MQ_DATA_CONV_EXIT ve skriptu MQ_DATA_CONV_EXITjsou popsány některé další podmínky.

Podrobnosti o volání MQGET najdete v tématu MQGET . Uživatelské procedury pro převod dat nemohou používat jiná volání MQI, než je MQXCNVC.

A new copy of the exit is loaded when an application attempts to retrieve the first message that uses that *Format* since the application connected to the queue manager. Je-li správce front vyřazen dříve načtenou kopií, může být také nová kopie načtena také v jiných časech.

Uživatelská procedura pro převod dat se spustí v prostředí, jako je tomu u programu, který vydal volání MQGET. Stejně jako uživatelské aplikace může být program MCA (agent kanálu zpráv) odesílající zprávy do cílového správce front, který nepodporuje převod zpráv. Prostředí zahrnuje adresní prostor a profil uživatele, kde je to vhodné. Uživatelská procedura nemůže ohrozit integritu správce front, protože není spuštěna v prostředí správce front.

Zápis uživatelské procedury pro převod dat pro produkt WebSphere MQ v systémech UNIX and Linux

Informace o krocích, které je třeba vzít v úvahu při zápisu uživatelských programů pro převod dat pro produkt WebSphere MQ na systémy UNIX and Linux .

Postupujte takto:

1. Pojmenujte svůj formát zprávy. Název se musí vejít do pole *Format* deskriptoru MQMD a musí být zadán velkými písmeny, například MYFORMAT. Název *Format* nesmí obsahovat úvodní mezery. Koncové mezery jsou ignorovány. Název objektu nesmí mít více než osm nemezerových znaků, protože *Format* je dlouhé pouze osm znaků. Nezapomeňte použít tento název pokaždé, když odešlete zprávu.

   Je-li uživatelská procedura pro převod dat použita v prostředí s podprocesy, musí za něj následovat objekt loadable, aby indikoval, že se jedná o verzi s podporou podprocesů.
2. Vytvořte strukturu pro znázornění vaší zprávy. Příklad naleznete v tématu Platná syntaxe .
3. Spusťte tuto strukturu pomocí příkazu `crtmqcvx` a vytvořte fragment kódu pro váš výstup pro převod dat.

   Funkce generované příkazem `crtmqcvx` používají makra, která předpokládají, že všechny struktury jsou sbalené; jejich změnu, pokud se nejedná o tento případ.
4. Zkopírujte dodaný zdrojový soubor kostry, přejmenujte jej na název vašeho formátu zprávy, který jste nastavili v kroku “1” na stránce 401. Zdrojový soubor kostry a kopie jsou určeny pouze ke čtení.

   Zdrojový soubor kostry se nazývá amqsvfc0.c.
5. V produktu WebSphere MQ for AIXje také dodán soubor exportu kostry s názvem amqsvfc.exp . Okopírujte tento soubor, přejmenujte jej na MYFORMAT.EXP.
6. Kostra obsahuje ukázkový soubor záhlaví amqsvmha.h, v adresáři *MQ_INSTALLATION_PATH*/inc, kde *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován. Ujistěte se, že vaše cesta Include ukazuje na tento adresář k vyzvednutí tohoto souboru.

   Soubor amqsvmha.h obsahuje makra, která jsou použita kódem generovaným příkazem `crtmqcvx` . Pokud struktura, která má být převedená, obsahuje znaková data, tato makra volají aplikaci MQXCNVC.
7. Najděte následující rámečky komentáře ve zdrojovém souboru a vložte kód podle popisu:
   a. Směrem ke konci zdrojového souboru, rámeček komentáře začíná znaky:

      ```
      /* Insert the functions produced by the data-conversion exit */
      ```

      Zde vložte fragment kódu vygenerovaný v kroku “3” na stránce 401.
   b. V blízkosti středu zdrojového souboru začíná rámeček s komentářem:

      ```
      /* Insert calls to the code fragments to convert the format's */
      ```

      Po této akci bude následovat komentář k funkci `ConverttagSTRUCT`.

      Změňte název funkce na název funkce, kterou jste přidali v kroku “7.a” na stránce 401. Chcete-li aktivovat funkci, odeberte znaky komentáře. Pokud existuje několik funkcí, vytvořte volání pro každou z nich.
   c. V blízkosti začátku zdrojového souboru začíná rámeček s komentářem:

      ```
      /* Insert the function prototypes for the functions produced by */
      ```

Zde vložte příkazy prototypu funkce pro funkce přidané v kroku “3” na stránce 401 výše.

8. Zkompilujte svou uživatelskou proceduru jako sdílenou knihovnu s použitím volby MQStart jako vstupního bodu. Chcete-li to provést, viz “Kompilování dat pro převod dat na systémech UNIX and Linux” na stránce 402.
9. Umístěte výstup do výstupního adresáře. Výchozí výstupní adresář je `/var/mqm/exits` pro 32 bitové systémy a `/var/mqm/exits64`pro 64bitové systémy. Tyto adresáře můžete změnit v souboru qm.ini nebo mqclient.ini . Tato cesta může být nastavena pro každého správce front a uživatelská procedura je vyhledána pouze v této cestě nebo v cestách.

Poznámka:

1. Pokud produkt `crtmqcvx` používá zabalené struktury, všechny aplikace produktu WebSphere MQ musí být zkompilovány tímto způsobem.
2. Ukončovací programy konverze dat musí být reentrantní.
3. MQXCNVC je *jediné* volání MQI, které lze vydat z uživatelské procedury pro převod dat.

Kompilování dat pro převod dat na systémech UNIX and Linux
Příklady toho, jak zkompilovat uživatelskou proceduru pro převod dat na systémech UNIX and Linux .

Na všech platformách je vstupní bod do modulu MQStart.

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

AIX

Zkompilujte zdrojový kód ukončení zadáním jednoho z následujících příkazů:

32bitové aplikace

Nevláknová

```
cc -e MQStart -bE:MYFORMAT.exp -bM:SRE -o /var/mqm/exits/MYFORMAT \
    MYFORMAT.c -IMQ_INSTALLATION_PATH/inc
```

Vláknové

```
xlc_r -e MQStart -bE:MYFORMAT.exp -bM:SRE -o /var/mqm/exits/MYFORMAT_r \
    MYFORMAT.c -IMQ_INSTALLATION_PATH/inc
```

64bitové aplikace

Nevláknová

```
cc -q64 -e MQStart -bE:MYFORMAT.exp -bM:SRE -o /var/mqm/exits64/MYFORMAT \
    MYFORMAT.c -IMQ_INSTALLATION_PATH/inc
```

Vláknové

```
xlc_r -q64 -e MQStart -bE:MYFORMAT.exp -bM:SRE -o /var/mqm/exits64/MYFORMAT_r \
    MYFORMAT.c -IMQ_INSTALLATION_PATH/inc
```

Platforma HP-UX Itanium

Zkompilujte a propojte zdrojový kód zadáním jedné z následujících sad příkazů:

32bitové aplikace

Nevláknová

Kompilace zdrojového kódu ukončení:

```
c89 +e +z -c -D_HPUX_SOURCE -o MYFORMAT.o MYFORMAT.c -IMQ_INSTALLATION_PATH/inc
```

Propojte objekt uživatelské procedury:

```
ld +b: -b MYFORMAT.o +ee MQStart -o  \
    /var/mqm/exits/MYFORMAT -L/usr/lib/hpux32
rm MYFORMAT.o
```

Vláknové

Kompilace zdrojového kódu ukončení:

```
c89 +e +z -c -D_HPUX_SOURCE -o MYFORMAT.o MYFORMAT.c -IMQ_INSTALLATION_PATH/inc
```

Propojte objekt uživatelské procedury:

```
ld +b: -b MYFORMAT.o +ee MQStart -o  \
    /var/mqm/exits/MYFORMAT_r -L/usr/lib/hpux32 \
    -lpthread
rm MYFORMAT.o
```

64bitové aplikace

Nevláknová

Kompilace zdrojového kódu ukončení:

```
 c89 +DD64 +e +z -c -D_HPUX_SOURCE -o MYFORMAT.o MYFORMAT.c -IMQ_INSTALLATION_PATH/inc
```

Propojte objekt uživatelské procedury:

```
ld -b MYFORMAT.o +ee MQStart \
    -o  /var/mqm/exits64/MYFORMAT \
    -L/usr/lib/hpux64
rm MYFORMAT.o
```

Vláknové

Kompilace zdrojového kódu ukončení:

```
c89 +DD64 +e +z -c -D_HPUX_SOURCE -o MYFORMAT.o MYFORMAT.c -IMQ_INSTALLATION_PATH/inc
```

Propojte objekt uživatelské procedury:

```
ld -b MYFORMAT.o +ee MQStart  \
    -o  /var/mqm/exits64/MYFORMAT_r  \
    -L/usr/lib/hpux64 -lpthread
rm MYFORMAT.o
```

Linux

Zkompilujte zdrojový kód ukončení zadáním jednoho z následujících příkazů:

31bitové aplikace

Nevláknová

```
 gcc -m31 -shared -fPIC -o /var/mqm/exits/MYFORMAT MYFORMAT.c \
      -IMQ_INSTALLATION_PATH/inc
```

Vláknové

```
gcc -m31 -shared -fPIC -o /var/mqm/exits/MYFORMAT_r MYFORMAT.c
     -IMQ_INSTALLATION_PATH/inc
```

32bitové aplikace

Nevláknová

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/MYFORMAT MYFORMAT.c
     -IMQ_INSTALLATION_PATH/inc
```

Vláknové

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/MYFORMAT_r MYFORMAT.c
     -IMQ_INSTALLATION_PATH/inc
```

64bitové aplikace

Nevláknová

```
gcc -m64 -shared -fPIC -o /var/mqm/exits64/MYFORMAT MYFORMAT.c
     -IMQ_INSTALLATION_PATH/inc
```

Vláknové

```
gcc -m64 -shared -fPIC -o /var/mqm/exits64/MYFORMAT_r MYFORMAT.c
     -IMQ_INSTALLATION_PATH/inc
```

Solaris

Zkompilujte zdrojový kód ukončení zadáním jednoho z následujících příkazů:

32bitové aplikace

Platforma SPARC

```
cc -xarch=v8plus -KPIC -mt -G -o /var/mqm/exits/MYFORMAT \
   MYFORMAT.c -IMQ_INSTALLATION_PATH/inc -R/usr/lib/32 -lsocket -lnsl -ldl
```

Platformax86-64

```
cc -xarch=386 -KPIC -mt -G -o /var/mqm/exits/MYFORMAT \
   MYFORMAT.c -IMQ_INSTALLATION_PATH/inc -R/usr/lib/32 -lsocket -lnsl -ldl
```

64bitové aplikace

Platforma SPARC

```
cc -xarch=v9 -KPIC -mt -G -o /var/mqm/exits64/MYFORMAT \
   MYFORMAT.c -IMQ_INSTALLATION_PATH/inc -R/usr/lib/64 -lsocket -lnsl -ldl
```

Platformax86-64

```
cc -xarch=amd64 -KPIC -mt -G -o /var/mqm/exits64/MYFORMAT \
   MYFORMAT.c -IMQ_INSTALLATION_PATH/inc -R/usr/lib/64  -lsocket -lnsl -ldl
```

Zápis uživatelské procedury pro převod dat pro produkt WebSphere MQ for Windows

Informace o krocích, které je třeba vzít v úvahu při zápisu uživatelských programů pro převod dat pro produkt WebSphere MQ for Windows.

Postupujte takto:

1. Pojmenujte svůj formát zprávy. Název se musí vejít do pole *Format* MQMD. Název *Format* nesmí obsahovat úvodní mezery. Koncové mezery jsou ignorovány. Název objektu nesmí mít více než osm nemezerových znaků, protože *Format* je dlouhé pouze osm znaků.

   Soubor .DEF s názvem amqsvfcn.def je také dodán v adresáři ukázek, *MQ_INSTALLATION_PATH*\Tools\C\Samples. *MQ_INSTALLATION_PATH* je adresář, kde je

nainstalován produkt WebSphere MQ . Vezměte kopii tohoto souboru a přejmenujte ji, například na MYFORMAT.DEF. Ujistěte se, že název knihovny DLL, která se vytvoří, a jméno uvedené v souboru MYFORMAT.DEF jsou stejné. Přepište název FORMAT1 v souboru MYFORMAT.DEF s novým názvem formátu.

Nezapomeňte použít tento název pokaždé, když odešlete zprávu.

2. Vytvořte strukturu pro znázornění vaší zprávy. Příklad naleznete v tématu Platná syntaxe .
3. Spusťte tuto strukturu pomocí příkazu `crtmqcvx` a vytvořte fragment kódu pro váš výstup pro převod dat.

   Funkce generované příkazem CRTMQCVX používají makra, která jsou zapsána za předpokladu, že jsou všechny struktury sbaleny; jejich změny, pokud se nejedná o tento případ.
4. Zkopírujte dodaný zdrojový soubor kostry, amqsvfc0.c, přejmenujte jej na název vašeho formátu zprávy, který jste nastavili v kroku “1” na stránce 404.

   amqsvfc0.c je v *MQ_INSTALLATION_PATH*`\Tools\C\Samples` , kde *MQ_INSTALLATION_PATH* je adresář, kde je nainstalován produkt WebSphere MQ . (Výchozí instalační adresář je `C:\Program Files\IBM\WebSphere MQ`.)

   Kostra obsahuje ukázkový soubor záhlaví amqsvmha.h v adresáři *MQ_INSTALLATION_PATH*`\Tools\C\include` . Ujistěte se, že vaše cesta Include ukazuje na tento adresář k vyzvednutí tohoto souboru.

   Soubor amqsvmha.h obsahuje makra, která jsou použita kódem generovaným příkazem CRTMQCVX. Pokud struktura, která má být převedená, obsahuje znaková data, tato makra volají aplikaci MQXCNVC.
5. Najděte následující rámečky komentáře ve zdrojovém souboru a vložte kód podle popisu:

   a. Směrem ke konci zdrojového souboru, rámeček komentáře začíná znaky:

   ```
   /* Insert the functions produced by the data-conversion exit */
   ```

   Zde vložte fragment kódu vygenerovaný v kroku “3” na stránce 405.

   b. V blízkosti středu zdrojového souboru začíná rámeček s komentářem:

   ```
   /* Insert calls to the code fragments to convert the format's */
   ```

   Po této akci bude následovat komentář k funkci `ConverttagSTRUCT`.

   Změňte název funkce na název funkce, kterou jste přidali v kroku “5.a” na stránce 405. Chcete-li aktivovat funkci, odeberte znaky komentáře. Pokud existuje několik funkcí, vytvořte volání pro každou z nich.

   c. V blízkosti začátku zdrojového souboru začíná rámeček s komentářem:

   ```
   /* Insert the function prototypes for the functions produced by */
   ```

   Zde vložte příkazy prototypu funkce pro funkce přidané v kroku “3” na stránce 405.
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 WebSphere MQ .
7. Vydejte příkazový soubor pro kompilaci uživatelské procedury jako soubor DLL.

8. Umístěte výstup do podadresáře exit pod datovým adresářem produktu WebSphere MQ . Výchozí adresář pro instalaci vašich uživatelských procedur na 32bitových systémech je *MQ_DATA_PATH*\Exits a pro 64bitové systémy je *MQ_DATA_PATH*\Exits64

   Cesta použitá k vyhledání uživatelských procedur pro převod dat je uvedena v registru. Složka registru je:

   ```
   HKEY_LOCAL_MACHINE\SOFTWARE\IBM\MQSeries\CurrentVersion\Configuration\ClientExitPa
   th\
   ```

   a klíč registru je: ExitsDefaultPath. Tato cesta může být nastavena pro každého správce front a uživatelská procedura je vyhledána pouze v této cestě nebo v cestách.

Poznámka:

1. Pokud CRTMQCVX používá sbalené struktury, všechny aplikace produktu WebSphere MQ je třeba v tomto směru kompilovat.
2. Ukončovací programy konverze dat musí být reentrantní.
3. MQXCNVC je *jediné* volání MQI, které lze vydat z uživatelské procedury pro převod dat.

Ukončit a přepnout soubory načtení na operačních systémech Windows

Procesy správce front produktu IBM WebSphere MQ for Windows Version 7.5 jsou 32bitové. V důsledku toho při použití 64bitových aplikací musí mít některé typy výstupu a zaváděcí soubory XA k dispozici také 32bitovou verzi, kterou může správce front používat. Je-li 32bitová verze výstupního nebo zaváděcího souboru přepínače XA vyžadována a není k dispozici, dojde k selhání příslušného volání nebo příkazu rozhraní API.

Dva atributy jsou podporovány v qm.ini file pro *ExitPath*. Jedná se o ExitsDefaultPath=*MQ_INSTALLATION_PATH*\exits a ExitsDefaultPath64=*MQ_INSTALLATION_PATH*\exits64. *MQ_INSTALLATION_PATH* Představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován. Pomocí těchto podmínek lze zajistit, že bude nalezena příslušná knihovna. Je-li uživatelská procedura použita v klastru WebSphere MQ , zajišťuje to také, že lze najít příslušnou knihovnu ve vzdáleném systému.

Následující tabulka obsahuje seznam různých typů souborů uživatelské procedury pro ukončení a přepínání a uvádí, zda jsou požadovány 32bitové nebo 64bitové verze, nebo obojí, podle toho, zda jsou používány 32bitové nebo 64bitové aplikace:

typy souboru	**32bitové aplikace**	**64bitové aplikace**
Uživatelská procedura napříč rozhraním API	32bitové	32 bitů a 64 bitů
Ukončení převodu dat	32bitové	64bitová
Uživatelské procedury kanálu serveru (všechny typy)	32bitové	32bitové
Uživatelské procedury kanálu klienta (všechny typy)	32bitové	64bitová
Ukončení instalovatelné služby	32bitové	32bitové
Modul trasování služby	32bitové	32 bitů a 64 bitů
Ukončení modulu WLM klastru	32bitové	32bitové
Publikování/odběr-ukončení směrování	32bitové	32bitové
Soubory načtení přepínače databáze	32bitové	32 bitů a 64 bitů

typy souboru	32bitové aplikace	64bitové aplikace
Knihovny produktu External Transaction Manager AX	32bitové	64bitová

Odkazování na definice připojení pomocí předání před připojením z úložiště

Klienty WebSphere MQ MQI lze konfigurovat tak, aby vyhledal úložiště za účelem získání definic připojení s použitím knihovny uživatelské procedury pro připojení k předpřipojení.

Úvod

Klientská aplikace se může připojit ke správci front pomocí tabulek CCDT (Client Channel Definition tabulek). Obecně je soubor CCDT umístěn na centrálním síťovém souborovém serveru a na něm klienti odkazují. Vzhledem k tomu, že je obtížné spravovat a spravovat různé klientské aplikace odkazující na soubor CCDT, je flexibilním přístupem ukládat definice klientů do globálního úložiště, jako je adresář LDAP, registr WebSphere a úložiště nebo jiné úložiště. Uložení definic připojení klienta v úložišti usnadňuje správu definic připojení klienta a aplikace mohou přistupovat ke správným a nejaktuálnějším definicím připojení klienta.

Během provádění volání MQCONN/X načte produkt IBM WebSphere MQ MQI client určenou aplikaci uživatelské procedury předběžného připojení a vyvolá uživatelskou funkci pro načtení definic připojení. Načtené definice připojení se pak použijí k navázání spojení se správcem front. Podrobnosti o knihovně uživatelské procedury a funkci k vyvolání jsou určeny v konfiguračním souboru mqclient.ini .

Syntaxe

`void MQ_PRECONNECT_EXIT` (*pExitParms*, *pQMgrName*, *ppConnectOpts*, *pCompCode*, *pReason*);

Parametry

pExitParms

Typ: vstup PMQNXP /výstup

Struktura konfiguračního parametru **`PreConnection`** .

Tato struktura je přidělována a udržována volajícím pro ukončení.

pQMgr

Typ: PMQCHAR vstupní/výstupní

Název správce front.

Na vstupu je tento parametr řetězec filtru dodávaný do volání rozhraní API MQCONN prostřednictvím parametru **`QMgrName`** . Toto pole může být prázdné, explicitní nebo obsahovat určité zástupné znaky. Pole je změněno uživatelskou procedurou. Při volání procedury MQXR_TERM je parametr nastaven na hodnotu Null.

ppConnectOtty

Typ: ppConnectOpts vstup/výstup

Volby, které řídí akci MQCONNX.

Jedná se o ukazatel na strukturu voleb připojení MQCNO, která řídí akci volání rozhraní API MQCONN. Při volání procedury MQXR_TERM je parametr nastaven na hodnotu Null. Klient MQI vždy poskytuje strukturu MQCNO pro ukončení i v případě, že ji aplikace původně neposkytovala. Pokud aplikace poskytuje strukturu MQCNO, klient vytvoří duplikát a předá jej do uživatelské procedury, kde je upravena. Klient si zachová vlastnictví objektu MQCNO.

MQCD, na které odkazuje objekt MQCNO, má přednost před jakoukoli definicí připojení poskytnutou prostřednictvím pole. Klient používá strukturu MQCNO k připojení ke správci front a ostatní jsou ignorovány.

pCompKód
Typ: PMQLONG vstupní/výstupní

Kód dokončení.

Ukazatel na MQLONG, který přijímá kód dokončení ukončení. Musí se jednat o jednu z následujících hodnot:

- `MQCC_OK` -Úspěšné dokončení.
- `MQCC_WARNING` -Varování (částečné dokončení)
- `MQCC_FAILED` -Volání se nezdařilo.

pReason
Typ: PMQLONG vstupní/výstupní

Kód určující kvalifikaci pCompCode.

Ukazatel na hodnotu MQLONG, která přijímá kód příčiny ukončení. Je-li kód dokončení MQCC_OK, jediná platná hodnota je:

- MQRC_NONE-(0, x '000') Chybí důvod k vytvoření sestavy.

Je-li kód dokončení MQCC_FAILED nebo MQCC_WARNING, může funkce uživatelské procedury nastavit pole s kódem příčiny na jakoukoli platnou hodnotu MQRC_ *.

Vyvolání jazyka C

```
void MQ_PRECONNECT_EXIT (&ExitParms, &QMgrName, &pConnectOpts, &CompCode, &Reason);
```

Parameter

```
PMQNXP  pExitParms   /*PreConnect exit parameter structure*/
PMQCHAR pQMgrName    /*Name of the queue manager*/
PPMQCNO ppConnectOpts/*Options controlling the action of MQCONNX*/
PMQLONG pCompCode    /*Completion code*/
PMQLONG pReason      /*Reason qualifying pCompCode*/
```

stanza PreConnect konfiguračního souboru klienta

Použijte sekci PreConnect ke konfiguraci uživatelské procedury PreConnect v souboru `mqclient.ini` .

Do sekce PreConnect mohou být zahrnuty následující atributy:

`Data=`*< URL>*
Adresa URL úložiště, kde jsou uloženy definice připojení. Například při použití serveru LDAP:

Data = ldap://myLDAPServer.com:389/cn=wmq,ou=ibm,ou=com

`Function=`*<myFunc>*
Název funkčního vstupního bodu do knihovny, která obsahuje výstupní kód PreConnect .

Definice funkce dodržuje prototyp uživatelské procedury PreConnect MQ_PRECONNECT_EXIT.

Maximální délka tohoto pole je MQ_EXIT_NAME_LENGTH.

`Module=`*< amqldapi>*
Název modulu obsahujícího kód ukončení rozhraní API.

Pokud toto pole obsahuje úplnou cestu k modulu, použije se tak, jak je.

`Sequence=`*< pořadové_číslo>*
Posloupnost, ve které je tato uživatelská procedura volána relativně k jiným uživatelským procedurám. Uživatelská procedura s nízkým pořadovým číslem se volá před ukončením s vyšším pořadovým číslem. Není třeba, aby sekvenční číslování vstupů bylo spojováno; posloupnost 1, 2, 3 má stejný výsledek jako posloupnost 7, 42, 1096. Tento atribut je nepodepsaná číselná hodnota.

V souboru `mqclient.ini` může být definováno více oddílů PreConnect . Pořadí zpracování každé procedury je určeno atributem Posloupnost stanzy.

Zápis a kompilace uživatelských procedur pro publikování

Chcete-li změnit obsah publikované zprávy před jejím přijetím odběrateli, můžete nakonfigurovat uživatelskou proceduru publikování ve správci front. Můžete také změnit záhlaví zprávy nebo nedoručit zprávu do odběru.

Uživatelské procedury publikování nejsou v systému z/OSpodporovány.

Pomocí uživatelské procedury pro publikování můžete zkontrolovat a změnit zprávy doručené odběratelům:

- Prozkoumat obsah zprávy publikované pro každého odběratele
- Upravit obsah zprávy publikované pro každého odběratele
- Změnit frontu, do níž je zpráva vložena
- Zastavit doručení zprávy odběrateli

Psaní uživatelské procedury publikování

Kroky uvedené v tématu “Uživatelské procedury pro psaní a kompilaci a instalovatelné služby” na stránce 358vám pomohou při zápisu a kompilaci vaší uživatelské procedury.

Poskytovatel uživatelské procedury pro publikování definuje, co bude uživatelská procedura dělat. Ukončení se však musí podřídit pravidlům definovaným v souboru MQPSXP.

Produkt WebSphere MQ neposkytuje implementaci vstupního bodu MQ_PUBLISHER_EXIT. Poskytuje deklaraci typedef jazyka C. Použijte typedef k deklarování parametrů pro uživatelskou proceduru správně. Následující příklad ukazuje, jak použít deklaraci typedef:

```
#include "cmqec.h"

MQ_PUBLISH_EXIT MyPublishExit;

void MQENTRY MyPublishExit( PMQPSXP pExitParms,
                            PMQPBC  pPubContext,
                            PMQSBC  pSubContext )
{
  /* C language statements to perform the function of the exit */
}
```

Uživatelská procedura publikování se spustí v rámci procesu správce front jako výsledek následujících operací:

- Operace publikování, kde je zpráva doručena jednomu nebo více odběratelům.
- Odebírat operaci, kde je doručena jedna nebo více zachovaných zpráv
- Operace Požadavek na odběr, ve které je doručena jedna nebo více zachovaných zpráv

Je-li uživatelská procedura publikování volána pro připojení, je nastavena první hodnota, která se nazývá *ExitReason* , je nastavena na hodnotu MQXR_INIT . Než se připojení odpojí po použití uživatelské procedury publikování, je uživatelská procedura volána s kódem *ExitReason* MQXR_TERM.

Je-li uživatelská procedura publikování konfigurována, ale nelze ji načíst při spuštění správce front, jsou pro správce front zakázány operace publikování/odběru zpráv. Před povolením systému zpráv publikování/odběru je třeba opravit problém nebo restartovat správce front.

Každé připojení produktu WebSphere MQ , které vyžaduje ukončení publikování, může selhat při načtení nebo inicializaci uživatelské procedury. Pokud se ukončení nepodaří načíst nebo inicializovat, operace publikování/odběru, které vyžadují uživatelskou proceduru publikování, jsou pro toto připojení zakázány. Operace se nezdaří s kódem příčiny WebSphere MQ MQRC_PUBLISH_EXIT_ERROR.

Kontext, ve kterém je volána uživatelská procedura publikování, je připojení aplikace ke správci front. Oblast uživatelských dat je spravována správcem front pro každé připojení, které provádí operace

publikování. Uživatelská procedura může uchovávat informace v oblasti uživatelských dat pro každé připojení.

Uživatelská procedura publikování může používat některá volání MQI. Může používat pouze ta volání MQI, která manipulují s vlastnostmi zprávy. Volání jsou:

- MQBUFMH5
- MQCRTM
- MQDLTMH
- MQDLTMP
- MQMBUF
- MQINQMP
- MQSETMP

Pokud uživatelská procedura publikování změní cílového správce front nebo název fronty, neprovádí se žádná nová kontrola oprávnění.

Kompilování uživatelské procedury publikování

Uživatelská procedura publikování je dynamicky načtenou knihovnou; lze ji považovat za uživatelskou proceduru kanálu. Informace o kompilaci uživatelských procedur viz "Uživatelské procedury pro psaní a kompilaci a instalovatelné služby" na stránce 358.

Ukázka uživatelské procedury publikování

Ukázkový ukončovací program se nazývá `amqspse0.c`. Do souboru protokolu bude zapsána jiná zpráva v závislosti na tom, zda byla uživatelská procedura volána pro operace inicializace, publikování nebo ukončení. Také demonstruje použití pole uživatelské oblasti pro ukončení k tomu, aby bylo možné vhodně přidělit a uvolnit paměť.

Konfigurace uživatelských procedur publikování

Chcete-li konfigurovat uživatelskou proceduru publikování, musíte definovat určité atributy.

V systémech Windows a Linux můžete k definování atributů použít průzkumníka WebSphere MQ . Atributy jsou definovány na stránce vlastností správce front v části Publikování/odběr.

Chcete-li nakonfigurovat uživatelskou proceduru publikování v souboru `qm.ini` na systémech UNIX a Linux , vytvořte oddíl s názvem `PublishSubscribe`. Stanza `PublishSubscribe` má následující atributy:

PublishExitPath=[path]|*module_name*
Název modulu a cesta obsahující kód uživatelské procedury publikování. Maximální délka tohoto pole je `MQ_EXIT_NAME_LENGTH`. Předvolba je žádná uživatelská procedura publikování.

PublishExitFunction=*function_name*
Název vstupního bodu funkce do modulu, který obsahuje výstupní kód publikování. Maximální délka tohoto pole je `MQ_EXIT_NAME_LENGTH`.

PublishExitData=*string*
Pokud správce front volá uživatelskou proceduru pro publikování, předá strukturu `MQPSXP` jako vstup. Data zadaná pomocí atributu *PublishExitData* se poskytují v poli *ExitData* struktury. Řetězec může mít délku až `MQ_EXIT_DATA_LENGTH` znaků. Předvolba je 32 prázdných znaků.

Zápis a kompilace uživatelských procedur pracovní zátěže klastru

Chcete-li přizpůsobit správu pracovní zátěže klastrů, zapište výstupní program pracovní zátěže klastru. Při směrování zpráv můžete při směrování zpráv brát v úvahu náklady na použití kanálu v různých denních dobách nebo při směrování zprávy. Jedná se o faktory, které nejsou brány v úvahu pro standardní algoritmus správy pracovní zátěže.

Ve většině případů je algoritmus správy pracovní zátěže dostatečný pro vaše potřeby. Nicméně, abyste mohli poskytnout svůj vlastní uživatelský program k přizpůsobení správy pracovní zátěže, produkt WebSphere MQ obsahuje uživatelskou proceduru a uživatelská procedura pracovní zátěže klastru.

Můžete mít některé konkrétní informace o své síti nebo zprávách, které byste mohli použít k ovlivnění vyrovnávání pracovní zátěže. Možná víte, které jsou vysokokapacitní kanály nebo levné přenosové cesty k síti, nebo můžete chtít směrovat zprávy v závislosti na jejich obsahu. Můžete se rozhodnout, zda chcete napsat uživatelský program pracovní zátěže klastru, nebo použít některý z nich dodaný třetí stranou.

Uživatelská procedura pracovní zátěže klastru je volána při přístupu ke frontě klastru. Nazývá se to MQOPEN, MQPUT1 a MQPUT.

Cílový správce front vybraný v době MQOPEN je pevný, je-li zadán MQOO_BIND_ON_OPEN . V tomto případě je uživatelská procedura spuštěna pouze jednou.

Pokud není správce cílové fronty opraven v době MQOPEN , je cílový správce front vybrán v době volání příkazu MQPUT . Pokud cílový správce front není k dispozici nebo selže, zatímco zpráva je stále v přenosové frontě, je uživatelská procedura volána znovu. Je vybrán nový cílový správce front. Pokud kanál zpráv selže během přenosu zprávy a je vrácena zpráva, je vybrán nový cílový správce front.

Na jiných platformách než z/OSnačte správce front novou uživatelskou proceduru pracovní zátěže klastru při příštím spuštění správce front.

Pokud definice správce front neobsahuje název ukončovacího programu pracovní zátěže klastru, nebude uživatelská procedura pracovní zátěže klastru volána.

Do ukončení pracovní zátěže klastru ve struktuře výstupního parametru MQWXPse předávají různé údaje:

- Struktura definice zprávy, MQMD.
- Parametr délky zprávy.
- Kopie zprávy nebo část zprávy.

Pokud používáte produkt CLWLMode=FASTna jiných platformách nežz/OS , načte každý proces operačního systému svou vlastní kopii uživatelské procedury. Různá připojení ke správci front mohou způsobit vyvolání různých kopií uživatelské procedury. Je-li uživatelská procedura spuštěna ve výchozím bezpečném režimu CLWLMode=SAFE, je jedna kopie uživatelské procedury spuštěna ve svém vlastním samostatném procesu.

Zápis uživatelských procedur pracovní zátěže klastru

U jiných platforem než z/OSnesmí uživatelská procedura pracovní zátěže klastru používat volání MQI. V jiných ohledech jsou pravidla pro zápis a kompilaci ukončovacích programů pracovní zátěže klastru podobná pravidlům, která se používají pro ukončovací programy kanálu. Postupujte podle kroků uvedených v tématu “Uživatelské procedury pro psaní a kompilaci a instalovatelné služby” na stránce 358a použijte ukázkový program “Ukázková uživatelská procedura pracovní zátěže klastru” na stránce 411 , který vám pomůže s napsáním a zkompilací vaší uživatelské procedury.

Další informace o uživatelských procedurách kanálů naleznete v části “Psaní programů výstupních bodů kanálu” na stránce 383.

Konfigurace uživatelských procedur pracovní zátěže klastru

Pracovní zátěž klastru se ukončí v definici správce front zadáním atributu uživatelské procedury pracovní zátěže klastru u příkazu ALTER QMGR . Příklad:

```
ALTER QMGR CLWLEXIT(myexit)
```

Ukázková uživatelská procedura pracovní zátěže klastru

Produkt WebSphere MQ obsahuje ukázkový výstupní program pracovní zátěže klastru. Vzorek můžete zkopírovat a použít jako základ pro své vlastní programy.

Na platformách jiných než z/OS

Ukázkový uživatelský program pracovní zátěže klastru se dodává v C a nazývá se `amqswlm0.c`. Je možné jej nalézt v:

Tabulka 57. Umístění ukázkového výstupního programu pracovní zátěže klastru (ne v systému z/OS)	
Platforma	**Cesta k souboru**
AIX, HP-UX, Sun Solaris	*MQ_INSTALLATION_PATH*`/samp`
Windows	*MQ_INSTALLATION_PATH*`\Tools\c\Samples`

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

Tato ukázková uživatelská procedura směruje všechny zprávy do konkrétního správce front, pokud nebude tento správce front nedostupný. Reaguje na selhání správce front přesměrováním zpráv do jiného správce front.

Označit, do kterého správce front se mají zasílat zprávy. Zadejte název přijímacího kanálu klastru do atributu `CLWLDATA` v definici správce front. Příklad:

```
ALTER QMGR CLWLDATA('my-cluster-name.my-queue-manager')
```

Chcete-li tuto proceduru povolit, zadejte její úplnou cestu a název do atributu `CLWLEXIT` :

Na systémech UNIX and Linux :

```
ALTER QMGR CLWLEXIT('path/amqswlm(clwlFunction)')
```

V systému Windows:

```
ALTER QMGR CLWLEXIT('path\amqswlm(clwlFunction)')
```

Nyní místo použití dodaného algoritmu správy pracovní zátěže zavolá produkt WebSphere MQ k přesměrování všech zpráv do zvoleného správce front.

Sestavení aplikace IBM WebSphere MQ

Tyto informace použijte k seznámení se s aplikací produktu IBM WebSphere MQ na různých platformách.

Vytváření vaší aplikace v systému AIX

Publikace AIX popisují, jak sestavit spustitelné aplikace z programů, které napíšete.

Toto téma popisuje další úlohy a změny standardních úloh, které je třeba provést při sestavování produktu WebSphere MQ pro aplikace AIX pro spuštění v systému AIX. Volby C, C++ a COBOL jsou podporovány. Informace o přípravě programů C + +, viz Použití C++.

Úlohy, které je třeba provést při vytváření spustitelné aplikace pomocí produktu WebSphere MQ for AIX , se liší podle programovacího jazyka, ve kterém je napsán váš zdrojový kód. Kromě kódování volání MQI ve vašem zdrojovém kódu musíte přidat příslušné příkazy jazyka pro zahrnutí produktu WebSphere MQ for AIX do souborů pro jazyk, který používáte. Seznamte se s obsahem těchto souborů. Viz “Soubory definic dat produktu IBM WebSphere MQ” na stránce 77 , kde získáte úplný popis.

Při spuštění serveru se závitem nebo klientskými aplikacemi s podporou podprocesů nastavte proměnnou prostředí AIXTHREAD_SCOPE = S.

Příprava programů v jazyce C v systému AIX

Toto téma obsahuje informace o propojování knihoven nezbytných pro přípravu programů v jazyce C v systému AIX.

Předkompilované programy C jsou dodávány v adresáři *MQ_INSTALLATION_PATH*/samp/bin . Použijte kompilátor ANSI a spusťte následující příkazy. Další informace o programování 64bitových aplikací najdete v tématu Koding standardů na 64bitových platformách.

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

Pro 32 bitové aplikace:

```
  $ xlc_r -o amqsput_32 amqsput0.c -IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -lmqm
```

kde `amqsput0` je ukázkový program.

Pro 64bitové aplikace:

```
  $ xlc_r -q64 -o amqsput_64 amqsput0.c -IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib64
-lmqm
```

kde `amqsput0` je ukázkový program.

Používáte-li kompilátor C/C++ VisualAge pro programy C + +, musíte zahrnout volbu `-q namemangling=v5` pro získání všech symbolů WebSphere MQ při propojování knihoven.

Chcete-li používat programy v počítači, který má instalován pouze klient WebSphere MQ MQI pro produkt AIX , znovu zkompilujte programy a propojte je s knihovnou klienta (`-lmqic`).

Propojování knihoven

Potřebujete následující knihovny:

- Propojte své programy s příslušnou knihovnou poskytovanou produktem WebSphere MQ.

  V prostředí bez podprocesů se připojte k jedné z následujících knihoven:

Soubor knihovny	Typ programu/ukončení
libmqm.a	Server pro C
libmqic.a & libmqm.a	Klient pro C

  V prostředí s podprocesy se připojte k jedné z následujících knihoven:

Soubor knihovny	Typ programu/ukončení
libmqm_r.a	Server pro C
libmqic_r.a & libmqm_r.a	Klient pro C

  Chcete-li například sestavit jednoduchou aplikaci WebSphere MQ s podporou podprocesů z jedné kompilační jednotky, spusťte následující příkazy.

  Pro 32 bitové aplikace:

```
  $ xlc_r -o amqsputc_32_r amqsput0.c -IMQ_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 -IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib64
-lmqm_r
```

  kde `amqsput0` je ukázkový program.

Chcete-li používat programy v počítači, který má instalován pouze klient WebSphere MQ MQI pro produkt AIX , znovu zkompilujte programy a propojte je s knihovnou klienta (`-lmqic`).

Poznámka:

1. Pokud píšete instalovatelnou službu (viz Administrace pro další informace), musíte se odkázat na knihovnu `libmqmzf.a` v aplikaci bez podprocesů a na knihovnu `libmqmzf_r.a` v aplikaci se závitem.
2. Pokud vytváříte aplikaci pro externí koordinaci správce transakcí kompatibilní se standardem XA, jako např. IBM TXSeries, Encinanebo BEA Tuxedo, musíte se připojit k serveru `libmqmxa.a` (nebo `libmqmxa64.a` , pokud váš správce transakcí zachází s typem " long ' typu 64 bitů) a `libmqz.a` v aplikaci bez podprocesů a ke knihovnám `libmqmxa_r.a` (nebo `libmqmxa64_r.a`) a `libmqz_r.a` v aplikaci s podprocesy.
3. Musíte propojit důvěryhodné aplikace se závitem WebSphere MQ vlákny. Nicméně v jednom okamžiku může být připojeno pouze jedno vlákno v důvěryhodné aplikaci na systému WebSphere MQ v systémech UNIX and Linux .
4. Knihovny produktu WebSphere MQ je třeba propojit před všemi ostatními knihovnami produktu.

Příprava programů COBOL v produktu AIX

Tyto informace použijte při přípravě programů v jazyce COBOL v produktu AIX pomocí sady IBM COBOL Set a Micro Focus COBOL.

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

- 32bitové příručky pro kopírování v COBOLu jsou instalovány v následujícím adresáři:

```
MQ_INSTALLATION_PATH/inc/cobcpy32
```

  a symbolické odkazy jsou vytvořeny v:

```
MQ_INSTALLATION_PATH/inc
```

- 64bitové příručky pro kopírování v jazyce COBOL jsou instalovány v následujícím adresáři:

```
MQ_INSTALLATION_PATH/inc/cobcpy64
```

V následujících příkladech nastavte proměnnou prostředí **COBCPY** na:

```
MQ_INSTALLATION_PATH/inc/cobcpy32
```

pro 32bitové aplikace a:

```
MQ_INSTALLATION_PATH/inc/cobcpy64
```

pro 64bitové aplikace.

Musíte propojit svůj program s jedním z následujících souborů knihovny:

Soubor knihovny	Typ programu/ukončení
libmqmcb.a	Server pro COBOL (aplikace bez podprocesů)
libmqmcb_r.a	Server pro COBOL (podprocesová aplikace)

Soubor knihovny	Typ programu/ukončení
libmqicb.a	Klient pro COBOL (aplikace bez podprocesů
libmqicb_r.a	Klient pro COBOL (podprocesová aplikace)

V závislosti na programu můžete použít kompilátor IBM COBOL Set nebo Micro Focus COBOL compiler.

- Programy začínající `amqm` jsou vhodné pro kompilátor Micro Focus COBOL a
- Programy začínající `amq0` jsou vhodné pro kompilátor.

Příprava programů v jazyce COBOL pomocí sady IBM COBOL Set for AIX

Ukázkové programy v jazyku COBOL jsou dodávány s produktem IBM WebSphere MQ. Chcete-li zkompilovat takový program, zadejte příslušný příkaz z následujícího seznamu:

32bitová serverová aplikace bez podprocesů

```
$ cob2 -o amq0put0 amq0put0.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb -qLIB \
-I<COBCPY>
```

32bitová klientská aplikace bez podprocesů

```
$ cob2 -o amq0put0 amq0put0.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb -qLIB \
-I<COBCPY>
```

32bitová serverová serverová aplikace

```
$ cob2_r -o amq0put0 amq0put0.cbl -qTHREAD -L MQ_INSTALLATION_PATH/lib \
-lmqmcb_r -qLIB -I<COBCPY>
```

Aplikace klienta s 32bitovým podprocesem

```
$ cob2_r -o amq0put0 amq0put0.cbl -qTHREAD -L MQ_INSTALLATION_PATH/lib \
-lmqicb_r -qLIB -I<COBCPY>
```

64bitová serverová aplikace bez podprocesů

```
$ cob2 -o amq0put0 amq0put0.cbl -q64 -L MQ_INSTALLATION_PATH/lib - lmqmcb \
-qLIB -I<COBCPY>
```

Aplikace klienta s 64bitovým nevláknem

```
$ cob2 -o amq0put0 amq0put0.cbl -q64 -L MQ_INSTALLATION_PATH/lib - lmqicb \
-qLIB -I<COBCPY>
```

64bitová serverová serverová aplikace

```
$ cob2_r -o amq0put0 amq0put0.cbl -q64 -qTHREAD -L MQ_INSTALLATION_PATH/lib \
-lmqmcb_r -qLIB -I<COBCPY>
```

Aplikace klienta s 64bitovým vláknem

```
$ cob2_r -o amq0put0 amq0put0.cbl -q64 -qTHREAD -L MQ_INSTALLATION_PATH/lib \
-lmqicb_r -qLIB -I<COBCPY>
```

Příprava programů v jazyce COBOL pomocí Micro Focus COBOL

Před kompilací programu nastavte proměnné prostředí následujícím způsobem:

```
export COBCPY=<COBCPY>
export LIBPATH=MQ_INSTALLATION_PATH/lib:$LIBPATH
```

Chcete-li zkompilovat 32bitového programu COBOL pomocí Micro Focus COBOL, zadejte:

- Server pro COBOL

```
$ cob32 -xvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb
```

- Klient pro COBOL

```
$ cob32 -xvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb
```

- Threadovaný server pro COBOL

```
$ cob32 -xtvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb_r
```

- Klient s podporou podprocesů pro COBOL

```
$ cob32 -xtvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb_r
```

Chcete-li zkompilovat 64bitový program COBOL pomocí Micro Focus COBOL, zadejte:

- Server pro COBOL

```
$ cob64 -xvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcb
```

- Klient pro COBOL

```
$ cob64 -xvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicb
```

- Threadovaný server pro COBOL

```
$ cob64 -xtvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcb_r
```

- Klient s podporou podprocesů pro COBOL

```
$ cob64 -xtvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicb_r
```

kde `amqminqx` je ukázkový program.

Popis proměnných prostředí, které je třeba nastavit, najdete v dokumentaci Micro Focus COBOL.

Příprava aplikačních programů produktu CICS v systému AIX

Tyto informace použijte při přípravě programů CICS v produktu AIX.

K dispozici jsou moduly přepínače XA, které vám umožňují propojit produkt CICS s produktem IBM WebSphere MQ:

Tabulka 58. Základní kód pro aplikační programy produktu CICS v systému AIX: inicializační rutina XA		
Popis	**C (zdroj)**	**C (exec)-přidání do XAD.Stanza**
inicializační rutina XA	amqzscix.c	*amqzsc* - CICS pro AIX

Použijte předem sestavenou verzi zaváděcího souboru přepínače IBM WebSphere MQ *amqzsc* , který se dodává spolu s produktem.

Vždy propojte transakce jazyka C se zabezpečenými podprocesy IBM WebSphere MQ knihovny *libmqm_r.a.*, a vaše transakce v jazyce COBOL s knihovnou COBOL *libmqmcb_r.a.*.

Další informace o podpoře transakcí CICS v produktu Administracenaleznete.

Podpora TXSeries CICS

Produkt IBM WebSphere MQ v systému AIX podporuje TXSeries CICS pomocí rozhraní XA. Ujistěte se, že aplikace CICS jsou připojeny ke vláknové verzi knihoven IBM WebSphere MQ .

Programy produktu CICS můžete spouštět pomocí sady IBM COBOL Set for AIX nebo Micro Focus COBOL. Následující oddíly popisují rozdíl mezi spuštěnými programy CICS na sadě COBOL IBM pro AIX a Micro Focus COBOL.

Zápis programů WebSphere MQ , které jsou načteny do stejné oblasti CICS v jazyce C nebo COBOL. Nemůžete vytvořit kombinaci volání C a COBOL MQI do stejné oblasti CICS . Většina volání MQI ve druhém použitém jazyce se nezdařila s kódem příčiny `MQRC_HOBJ_ERROR`.

Příprava programů CICS COBOL pomocí produktu IBM COBOL Set for AIX

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

Chcete-li použít produkt IBM COBOL, proveďte následující kroky:

1. Exportovat následující proměnnou prostředí:

```
export LDFLAGS="-qLIB -bI:/usr/lpp/cics/lib/cicsprIBMCOB.exp  \
                -IMQ_INSTALLATION_PATH/inc -I/usr/lpp/cics/include  \
                -e _iwz_cobol_main  \
```

   kde `LIB` je direktiva kompilátoru.
2. Přeložte, zkompilujte a propojte program zadáním příkazu:

```
cicstcl -l IBMCOB <yourprog>.ccp
```

Příprava programů CICS COBOL pomocí Micro Focus COBOL

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

Chcete-li použít funkci Micro Focus COBOL, postupujte takto:

1. Přidejte modul knihovny běhové komponenty produktu IBM WebSphere MQ COBOL do knihovny běhového prostředí pomocí následujícího příkazu:

```
cicsmkcobol -L/usr/lib/dce -LMQ_INSTALLATION_PATH/lib  \
            MQ_INSTALLATION_PATH/lib/libmqmcbrt.o -lmqe_r
```

   Poznámka: With `cicsmkcobol`, IBM WebSphere MQ does not allow you to make MQI calls in the C programming language from your COBOL application.

   Pokud mají existující aplikace taková volání, doporučuje se tyto funkce přesunout z aplikací v jazyce COBOL do své vlastní knihovny, například `myMQ.so`. Po přesunu funkcí nezahrnujte knihovnu IBM WebSphere MQ `libmqmcbrt.o` při sestavování aplikace COBOL pro CICS.

   Navíc, pokud vaše aplikace v jazyku COBOL nevolá žádné volání COBOL MQI, nepropojujte `libmqmz_r` s `cicsmkcobol`.

   Tím se vytvoří soubor metod jazyka COBOL Micro Focus a povolí běhové knihovně jazyka COBOL produktu CICS volání IBM WebSphere MQ na systémech UNIX and Linux .

   Poznámka: Spusťte produkt `cicsmkcobol` pouze při instalaci jednoho z následujících produktů:

   - Nová verze nebo vydání Micro Focus COBOL
   - Nová verze nebo vydání CICS pro AIX
   - Nová verze nebo vydání všech podporovaných databázových produktů (pouze pro transakce v jazyce COBOL)
   - Nová verze nebo vydání IBM WebSphere MQ

2. Exportovat následující proměnnou prostředí:

```
COBCPY=MQ_INSTALLATION_PATH/inc  export COBCPY
```

3. Přeložte, zkompilujte a propojte program zadáním příkazu:

```
cicstcl -l COBOL -e <yourprog>.ccp
```

Příprava programů v jazyku CICS C

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

Sestavte programy CICS C pomocí standardních zařízení CICS :

1. Exportujte ***jeden*** z následujících proměnných prostředí:
   - LDFLAGS = "-L/*MQ_INSTALLATION_PATH*lib -lmqm_r" export LDFLAGS
   - USERLIB = "-L*MQ_INSTALLATION_PATH*lib -lmqm_r" export USERLIB
2. Přeložte, zkompilujte a propojte program zadáním příkazu:

```
cicstcl -l C amqscic0.ccs
```

Ukázková transakce CICS C

Ukázka zdroje C pro transakci produktu AIX IBM WebSphere MQ je k dispozici v rámci AMQSCIC0.CCS. Transakce čte zprávy z přenosové fronty SYSTEM.SAMPLE.CICS.WORKQUEUE na výchozím správci front a umístěte je do lokální fronty s názvem fronty, který je obsažen v záhlaví přenosu zprávy. Veškerá selhání se odesílají do fronty SYSTEM.SAMPLE.CICS.DLQ. Použijte ukázkový skript MQSC AMQSCIC0.TST pro vytvoření těchto front a ukázkových vstupních front.

Vytváření vaší aplikace v systému HP Integrity NonStop Server

Tyto informace popisují další úlohy a změny standardních úloh, které musíte provést při sestavování klienta produktu IBM WebSphere MQ pro aplikace produktu HP Integrity NonStop Server , které mají být spuštěny v rámci produktu HP Integrity NonStop Server.

Volby C, COBOL a pTAL jsou podporovány.

Záhlaví OSS a Guardian a veřejné knihovny

Poskytuje seznamy záhlaví OSS a Guardian a veřejných knihoven. Uvedeny jsou záhlaví OSS, veřejné spustitelné soubory OSS a veřejné importní knihovny, záhlaví Guardian a veřejný spustitelný soubor Guardian a veřejné importní knihovny.

"OSS hlavičky" na stránce 419

"Veřejné spustitelné a veřejné importní knihovny OSS" na stránce 420

"Záhlaví Guardian" na stránce 420

"Veřejné spustitelné a veřejné importní knihovny Guardian" na stránce 421

OSS hlavičky

Tabulka 59. OSS hlavičky

Objekt	Umístění	Popis
cmqbc.h	<mqinstall>/inc	IBM WebSphere MQ -hlavička jazyka C (OSS)
cmqc.h	<mqinstall>/inc	IBM WebSphere MQ -hlavička jazyka C (OSS)
cmqfc.h	<mqinstall>/inc	IBM WebSphere MQ -hlavička jazyka C (OSS)
cmqec.h	<mqinstall>/inc	IBM WebSphere MQ -hlavička jazyka C (OSS)
cmqpsc.h	<mqinstall>/inc	IBM WebSphere MQ -hlavička jazyka C (OSS)
cmqxc.h	<mqinstall>/inc	IBM WebSphere MQ -hlavička jazyka C (OSS)
cmqzc.h	<mqinstall>/inc	IBM WebSphere MQ -hlavička jazyka C (OSS)
cmqcobol.cpy	<mqinstall>/inc	IBM WebSphere MQ COBOL copybook (OSS)
cmqbt.tal	<mqinstall>/inc	záhlaví IBM WebSphere MQ pTAL (OSS)
cmqcft.tal	<mqinstall>/inc	záhlaví IBM WebSphere MQ pTAL (OSS)
cmqpst.tal	<mqinstall>/inc	záhlaví IBM WebSphere MQ pTAL (OSS)
cmqt.tal	<mqinstall>/inc	záhlaví IBM WebSphere MQ pTAL (OSS)
cmqxt.tal	<mqinstall>/inc	záhlaví IBM WebSphere MQ pTAL (OSS)

Veřejné spustitelné a veřejné importní knihovny OSS

Tabulka 60. Veřejné spustitelné a veřejné importní knihovny OSS

Objekt	Umístění	Popis
libmqic.so	<mqinstall>/bin	IBM WebSphere MQ veřejná spustitelná knihovna (OSS bez podprocesů)
libmqic_r.so	<mqinstall>/bin	IBM WebSphere MQ veřejná spustitelná knihovna (OSS multithreaded)
libmqic.so	<mqinstall>/lib	IBM WebSphere MQ veřejná knihovna importu (OSS bez podprocesů)
libmqic_r.so	<mqinstall>/lib	IBM WebSphere MQ veřejná knihovna importu (OSS multithreaded)
mqicb	<mqinstall>/lib	IBM WebSphere MQ veřejná knihovna importu pro COBOL (OSS)

Záhlaví Guardian

Tabulka 61. Záhlaví Guardian

Objekt	Umístění	Popis
cmqbch	<mqinstall>/inc/G	IBM WebSphere MQ -Hlavička jazyka C (Guardian)
cmqch	<mqinstall>/inc/G	IBM WebSphere MQ -Hlavička jazyka C (Guardian)
cmqfch	<mqinstall>/inc/G	IBM WebSphere MQ -Hlavička jazyka C (Guardian)
cmqech	<mqinstall>/inc/G	IBM WebSphere MQ -Hlavička jazyka C (Guardian)
cmqpsch	<mqinstall>/inc/G	IBM WebSphere MQ -Hlavička jazyka C (Guardian)
cmqxch	<mqinstall>/inc/G	IBM WebSphere MQ -Hlavička jazyka C (Guardian)
cmqzch	<mqinstall>/inc/G	IBM WebSphere MQ -Hlavička jazyka C (Guardian)
cmqcobol	<mqinstall>/inc/G	IBM WebSphere MQ zakladač COBOL (Guardian)
cmqbt	<mqinstall>/inc/G	záhlaví IBM WebSphere MQ pTAL (Guardian)
cmqcft	<mqinstall>/inc/G	záhlaví IBM WebSphere MQ pTAL (Guardian)
cmqpst	<mqinstall>/inc/G	záhlaví IBM WebSphere MQ pTAL (Guardian)

Tabulka 61. Záhlaví Guardian (pokračování)		
Objekt	**Umístění**	**Popis**
cmqt	<mqinstall>/inc/G	záhlaví IBM WebSphere MQ pTAL (Guardian)
cmqxt	<mqinstall>/inc/G	záhlaví IBM WebSphere MQ pTAL (Guardian)

Veřejné spustitelné a veřejné importní knihovny Guardian

Tabulka 62. Veřejné spustitelné a veřejné importní knihovny Guardian		
Objekt	**Umístění**	**Popis**
mqiová	<mqinstall>/bin/G	IBM WebSphere MQ veřejná spustitelná knihovna (Guardian)
mqicb	<mqinstall>/lib/G	IBM WebSphere MQ veřejná knihovna pro import pro COBOL (Guardian)

Příprava programů jazyka C v produktu HP Integrity NonStop Server

Toto téma obsahuje informace, které je třeba zvážit při přípravě programů v jazyce C v produktu HP Integrity NonStop Server spolu s příklady příkazů, které používáte při sestavování aplikací při použití kompilátoru OSS C a používáte-li kompilátor jazyka Guardian C.

Předkompilované C programy jsou dodávány v adresáři MQ_INSTALLATION_PATH/opt/mqm/samp/bin . Chcete-li sestavit ukázku ze zdrojového kódu, použijte kompilátor c89 .

Musíte propojit své programy s příslušnou knihovnou poskytnutou produktem IBM WebSphere MQ. Následující tabulka obsahuje seznam knihoven, na které musíte odkazovat, když připravujete programy v jazyce C na serveru HP Integrity NonStop Server.

Tabulka 63. . Knihovny odkazů produktu HP Integrity NonStop Server	
Knihovna	**Popis**
libmqic.so	OSS bez podprocesů
libmqic_r.so	OSS s podporou podprocesů
mqiová	Guardian

Nativní aplikace IBM WebSphere MQ s podporou více podprocesů musí používat funkci Posix User Threads (PUT). V tomto produktu není žádná podpora pro standardní Po6 Threads (SPT).

Sestavení aplikací pomocí kompilátoru OSS C

Tato sekce obsahuje příklady příkazů, které se používají k sestavení programů, které jsou zaměřeny buď na OSS nebo Guardian, když používáte kompilátor OSS.

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

Následující příklad sestaví aplikaci OSS klienta typu C bez podprocesů:

```
c89 -Wsystype=oss -o amqsputc amqsput0.c -IMQ_INSTALLATION_PATH/opt/mqm/inc
-LMQ_INSTALLATION_PATH/opt/mqm/lib -lmqic
```

Následující příklad sestaví aplikaci OSS s podporou podprocesů C s více vlákny:

```
c89 -Wsystype=oss -D_PUT_MODEL_ -o amqsputc amqsput0.c -IMQ_INSTALLATION_PATH/opt/mqm/inc
-LMQ_INSTALLATION_PATH/opt/mqm/lib -lmqic_r -lput
```

V následujícím příkladu je založena aplikace klienta Guardian C:

```
c89 -Wsystype=guardian -o /G/vol/subvol/amqsputc amqsput0.c -IMQ_INSTALLATION_PATH/opt/mqm/inc
-LMQ_INSTALLATION_PATH/opt/mqm/lib/G -lmqic
```

Sestavování aplikací pomocí kompilátoru Guardian C

Tento oddíl obsahuje příklady příkazů, které se používají k sestavování programů, které jsou zaměřeny na Guardian při použití kompilátoru Guardian.

MQ_INSTALLATION_PATH představuje svazek Guardian a podsvazek, ve kterém je nainstalován produkt IBM WebSphere MQ .

V následujícím příkladu je založena aplikace klienta Guardian C:

```
CCOMP /in AMQSPUT0/ AMQSPUTC;&
       runnable,systype guardian,nolist,&
       ssv0 "$system.system",&
       ssv1 "MQINSTALLATION_SUBVOL",&
       ld(-LMQINSTALLATION_SUBVOL -lmqic)
```

Příprava programů COBOL

Toto téma obsahuje informace, které je třeba zvážit při přípravě programů v jazyce C pro klienta produktu IBM WebSphere MQ pro produkt HP Integrity NonStop Server. Obsahuje příklady příkazů, které používáte při sestavování aplikací, když používáte kompilátor OSS ECOBOL a používáte-li kompilátor Guardian ECOBOL.

Chcete-li sestavit ukázku jazyka COBOL ze zdrojového kódu, použijte kompilátor ECOBOL.

Následující tabulka obsahuje seznam knihoven, které jsou potřeba při přípravě programů v jazyce COBOL v systému HP Integrity NonStop Server. Musíte propojit své programy s příslušnou knihovnou poskytnutou produktem IBM WebSphere MQ.

Tabulka 64. . Knihovny odkazů produktu HP Integrity NonStop Server

Knihovna	**Popis**
libmqic.so	OSS bez podprocesů
mqiová	Guardian

Při spuštění aplikace v jazyce COBOL, která se připojuje ke správci front, je třeba nejprve nastavit proměnnou *SAVE-ENVIRONMENT* na hodnotu ON. Chcete-li nastavit proměnnou *SAVE-ENVIRONMENT* na hodnotu ON:

- Pro OSS zadejte tento příkaz:

  ```
  export SAVE-ENVIRONMENT=ON
  ```

- Pro Guardian zadejte tento příkaz:

  ```
  param SAVE-ENVIRONMENT ON
  ```

Pokud nenastavíte proměnnou *SAVE-ENVIRONMENT* na hodnotu ON, když se aplikace pokusí o připojení ke správci front, dojde k selhání s kódem příčiny 2058 (080A) (RC2058): MQRC_Q_MGR_NAME_ERROR.

Sestavení aplikací pomocí kompilátoru OSS ECOBOL

Tato sekce obsahuje příklady příkazů, které se používají k sestavení programů, které jsou zaměřeny buď na OSS nebo Guardian, když používáte kompilátor OSS ECOBOL.

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

Následující příklad sestaví aplikaci OSS klienta COBOL:

```
ecobol -Wsystype=oss
       -Wcobol="ansi;port"
       -Wcobol="consult MQ_INSTALLATION_PATH/opt/mqm/lib/mqicb"
       -Wcopylib=MQ_INSTALLATION_PATH/opt/mqm/inc/cmqcobol.cpy
       -LMQ_INSTALLATION_PATH/opt/mqm/lib -lmqic
       -o amq0put0
       MQ_INSTALLATION_PATH/opt/mqm/samp/amq0put0.cbl
```

Následující příklad vytváří sestavení aplikace Guardian klienta COBOL:

```
ecobol -Wsystype=guardian
       -Wcobol="ansi;port;save all"
       -Wcobol="consult MQ_INSTALLATION_PATH/opt/mqm/lib/mqicb"
       -Wcopylib=MQ_INSTALLATION_PATH/opt/mqm/inc/cmqcobol.cpy
       -LMQ_INSTALLATION_PATH/opt/mqm/lib/G -lmqic
       -o amq0put0
       MQ_INSTALLATION_PATH/opt/mqm/samp/amq0put0.cbl
```

Sestavování aplikací pomocí kompilátoru Guardian ECOBOL

Tato sekce obsahuje příklady příkazů, které se používají k sestavování programů, které jsou cílem programu Guardian, když používáte kompilátor ECOBOL Guardian.

MQ_INSTALLATION_SUBVOL představuje svazek Guardian a podsvazek, ve kterém je nainstalován produkt IBM WebSphere MQ .

Následující příklad vytváří sestavení aplikace Guardian klienta COBOL:

```
ECOBOL /in MQSPUTL/ MQSPUT,MQINSTALLATION_SUBVOL.cmqcobol;
       call-shared;ansi;port;save all;nolist;runnable;
       consult MQINSTALLATION_SUBVOL.mqicb;
       eld(-LMQINSTALLATION_SUBVOL -lmqic)
```

Příprava programů pTAL

Naučte se sestavovat programy pTAL pro klienta IBM WebSphere MQ na platformě HP Integrity NonStop Server .

Chcete-li sestavit ukázku pTAL ze zdrojového kódu, použijte kompilátor EPTAL.

Poznámka:

- Aplikace pTAL IBM WebSphere MQ musí používat hlavní rutinu, která je zapsána buď v jazycích C, nebo COBOL.
- Aplikace pTAL lze sestavit pouze v Guardianu.

V následující tabulce je uvedena knihovna, která je potřebná při přípravě programů pTAL na serveru HP Integrity NonStop Server. Musíte propojit své programy s příslušnou knihovnou poskytnutou produktem IBM WebSphere MQ.

Tabulka 65. . Knihovna odkazů HP Integrity NonStop Server

Knihovna	**Popis**
mqiová	Guardian

Sestavování aplikací pomocí kompilátoru EPTAL Guardian

Tato sekce obsahuje příklady příkazů, které se používají k sestavování programů, které jsou zaměřeny na Guardian, když používáte kompilátor EPTAL Guardian.

MQINSTALLATION_SUBVOL představuje svazek Guardian a podsvazek, ve kterém je nainstalován produkt IBM WebSphere MQ .

Aplikace pTAL IBM WebSphere MQ musí používat hlavní rutinu, která je zapsána buď v jazycích C, nebo COBOL.

Následující příklad sestaví aplikaci klienta pTAL Guardian:

```
ASSIGN SSV0, $SYSTEM.SYSTEM
ASSIGN SSV1, MQINSTALLATION_SUBVOL

EPTAL /in MQINSTALLATION_SUBVOL.MQSPUTT/ MQSPUTO;nolist

CCOMP /in MQINSTALLATION_SUBVOL.MQSPTMC/ MQSPUT;
       runnable,systype guardian,extensions,nolist,
       ssv0 "$system.system",
       ssv1 "MQINSTALLATION_SUBVOL",
       eld(MQSPUTO -LMQINSTALLATION_SUBVOL -lmqic)
```

Sestavení aplikace v systému HP-UX

Tyto informace popisují další úlohy a změny standardních úloh, které musíte provést při sestavování produktu WebSphere MQ pro aplikace HP-UX , které se mají spustit v systému HP-UX.

Volby C, C++ a COBOL jsou podporovány. Informace o přípravě programů C + +, viz Použití C++.

Úlohy, které je třeba provést při vytváření spustitelné aplikace pomocí produktu WebSphere MQ for HP-UX , se liší podle programovacího jazyka, ve kterém je napsán váš zdrojový kód. Kromě kódování volání MQI ve vašem zdrojovém kódu musíte přidat příslušné příkazy jazyka pro zahrnutí produktu WebSphere MQ for HP-UX do souborů začlenění pro jazyk, který používáte. Seznamte se s obsahem těchto souborů. Úplný popis najdete v tématu “Soubory definic dat produktu IBM WebSphere MQ” na stránce 77 .

V celém tomto tématu používáme znak zpětného lomítka (\) k rozdělení dlouhých příkazů na více než jeden řádek. Nezadávejte tento znak; zadejte každý příkaz jako jednu řádku.

Příprava programů v jazyku C v systému HP-UX

Toto téma obsahuje informace, které je třeba zvážit při přípravě programů v jazyce C v systému HP-UX; s příklady pro platformu IA64 (IPF).

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, v němž je nainstalován produkt WebSphere MQ .

Pracujte ve svém normálním prostředí. Předkompilované C programy jsou dodávány v adresáři *MQ_INSTALLATION_PATH*/samp/bin .

Další informace o programování 64bitových aplikací najdete v tématu Kódování standardů na 64 bitových platformách.

Chcete-li používat zabezpečení SSL, musí být klienti WebSphere MQ MQI v systému HP-UX sestaveny pomocí podprocesů POSIX .

Několik příkladů, které je třeba zvážit:

- “Platforma IA64 (IPF)” na stránce 424
- “Propojování knihoven” na stránce 426

Platforma IA64 (IPF)

Příklady sestavení amqsput0, cliexit a srvexit na platformě IA64(IPF).

Následující příklad sestaví ukázkový program amqsput0 jako klientskou aplikaci v 32bitovém prostředí bez podprocesů:

```
c89 -Wl,+b,: +e -D_HPUX_SOURCE -o amqsputc_32 amqsput0.c -IMQ_INSTALLATION_PATH/inc
  -LMQ_INSTALLATION_PATH/lib -L/usr/lib/hpux32 -lmqic
```

Následující příklad sestaví ukázkový program amqsput0 jako klientskou aplikaci v 32bitovém prostředí s podporou podprocesů:

```
c89 -mt -Wl,+b,: +e -D_HPUX_SOURCE -o amqsputc_32_r amqsput0.c -IMQ_INSTALLATION_PATH/inc
  -LMQ_INSTALLATION_PATH/lib -L/usr/lib/hpux32 -lmqic_r -lpthread
```

Následující příklad sestaví ukázkový program amqsput0 jako klientskou aplikaci v 64bitovém prostředí 64bitového prostředí:

```
c89 +DD64 +e -D_HPUX_SOURCE -o amqsputc_64 amqsput0.c -IMQ_INSTALLATION_PATH/inc
  -LMQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -lmqic
```

Následující příklad sestaví ukázkový program amqsput0 jako klientskou aplikaci v 64bitovém prostředí s podporou podprocesů:

```
c89 -mt +DD64 +e -D_HPUX_SOURCE -o amqsputc_64_r amqsput0.c -IMQ_INSTALLATION_PATH/inc
  -LMQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -lmqic_r -lpthread
```

Následující příklad sestaví ukázkový program amqsput0 jako serverovou aplikaci v 32bitovém prostředí bez podprocesů:

```
c89 -Wl,+b,: +e -D_HPUX_SOURCE -o amqsput_32 amqsput0.c -IMQ_INSTALLATION_PATH/inc
  -LMQ_INSTALLATION_PATH/lib -L/usr/lib/hpux32 -lmqm
```

Následující příklad sestaví ukázkový program amqsput0 jako serverovou aplikaci v 32bitovém prostředí s podporou podprocesů:

```
c89 -mt -Wl,+b,: +e -D_HPUX_SOURCE -o amqsput_32_r amqsput0.c -IMQ_INSTALLATION_PATH/inc
  -LMQ_INSTALLATION_PATH/lib -L/usr/lib/hpux32 -lmqm_r -lpthread
```

Následující příklad sestaví ukázkový program amqsput0 jako serverovou aplikaci v 64bitovém prostředí bez podprocesů.

```
c89 +DD64 +e -D_HPUX_SOURCE -o amqsput_64 amqsput0.c -IMQ_INSTALLATION_PATH/inc
  -LMQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -lmqm
```

Následující příklad sestaví ukázkový program amqsput0 jako serverovou aplikaci v prostředí s 64bitovým 64bitovým vláknem:

```
c89 -mt +DD64 +e -D_HPUX_SOURCE -o amqsput_64_r amqsput0.c -IMQ_INSTALLATION_PATH/inc
  -LMQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -lmqm_r -lpthread
```

V následujícím příkladu je založena uživatelská procedura ukončení klienta v 32bitovém prostředí bez podprocesů:

```
c89 +e +z -c -D_HPUX_SOURCE -o cliexit.o cliexit.c -IMQ_INSTALLATION_PATH/inc
ld +b: -b cliexit.o +ee MQStart -o /var/mqm/exits/cliexit_32 -LMQ_INSTALLATION_PATH/lib \
-L/usr/lib/hpux32 -lmqic
```

V následujícím příkladu je založena uživatelská procedura ukončení klienta v 32bitovém prostředí s podporou podprocesů:

```
c89 -mt +e +z -c -D_HPUX_SOURCE -o cliexit.o cliexit.c -IMQ_INSTALLATION_PATH/inc
ld +b: -b cliexit.o +ee MQStart -o /var/mqm/exits/cliexit_32_r -LMQ_INSTALLATION_PATH/lib \
-L/usr/lib/hpux32 -lmqic_r -lpthread
```

Následující příklad sestaví uživatelskou proceduru pro ukončení klienta v 64bitovém prostředí bez podprocesů:

```
c89 +DD64 +e +z -c -D_HPUX_SOURCE -o cliexit.o cliexit.c -IMQ_INSTALLATION_PATH/inc
ld -b cliexit.o +ee MQStart -o /var/mqm/exits64/cliexit_64 \
-LMQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -lmqic
```

Následující příklad sestaví uživatelskou proceduru pro ukončení klienta v 64bitovém prostředí se závitovým prostředím:

```
c89 -mt +DD64 +e +z -c -D_HPUX_SOURCE -o cliexit.o cliexit.c -IMQ_INSTALLATION_PATH/inc
ld  -b cliexit.o +ee MQStart -o /var/mqm/exits/cliexit_64_r \
-LMQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -lmqic_r -lpthread
```

Následující příklad sestaví proceduru srvexit serveru v 32bitovém prostředí bez podprocesů se závity:

```
c89 +e +z -c -D_HPUX_SOURCE -o srvexit.o srvexit.c -IMQ_INSTALLATION_PATH/inc
ld +b: -b srvexit.o +ee MQStart -o /var/mqm/exits/srvexit_32 -LMQ_INSTALLATION_PATH/lib \
-L/usr/lib/hpux32 -lmqm
```

Následující příklad sestaví server srvexit serveru v 32bitovém prostředí s podporou podprocesů:

```
c89 -mt +e +z -c -D_HPUX_SOURCE -o srvexit.o srvexit.c -IMQ_INSTALLATION_PATH/inc
ld +b: -b srvexit.o +ee MQStart -o /var/mqm/exits/srvexit_32_r -LMQ_INSTALLATION_PATH/lib \
-L/usr/lib/hpux32 -lmqm_r -lpthread
```

Následující příklad sestaví proceduru srvexit serveru v 64bitovém prostředí 64bitového prostředí:

```
c89 +DD64 +e +z -c -D_HPUX_SOURCE -o srvexit.o srvexit.c
-IMQ_INSTALLATION_PATHMQ_INSTALLATION_PATH/inc
ld -b srvexit.o +ee MQStart -o /var/mqm/exits64/srvexit_64 \
-LMQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -lmqm
```

Následující příklad sestaví server srvexit serveru v 64bitovém prostředí s podporou podprocesů:

```
c89 -mt +DD64 +e +z -c -D_HPUX_SOURCE -o srvexit.o srvexit.c -IMQ_INSTALLATION_PATH/inc
ld  -b srvexit.o +ee MQStart -o /var/mqm/exits/srvexit_64_r \
-LMQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -lmqm_r -lpthread
```

Propojování knihoven

Je třeba propojit své programy s příslušnou knihovnou poskytovanou produktem WebSphere MQ.

Následující tabulka ukazuje, která knihovna se má použít v různých prostředích

Hardwarová platforma	Prostředí s podprocesy nebo bez podprocesů	Typ programu/ukončení	Soubor knihovny
IA64 (IPF)	Vláknové	Server & Klient pro C	libmqm_r.so
IA64 (IPF)	Vláknové	Klient pro C	libmqic_r.so
IA64 (IPF)	Nevláknová	Server & Klient pro C	libmqm.so
IA64 (IPF)	Nevláknová	Klient pro C	libmqic.so

Poznámka:

1. Pokud zapisujete instalovatelnou službu (viz Administrace pro další informace), musíte se připojit ke knihovně `libmqmzf.sl` .
2. Pokud vytváříte aplikaci pro externí koordinaci správce transakcí kompatibilní se standardem XA, jako např. IBM TXSeries Encinanebo BEA Tuxedo, musíte se připojit k serveru `libmqmxa.sl` (nebo `libmqmxa64.sl` , pokud váš správce transakcí zachází s typem " long ' typu 64 bitů) a `libmqz.sl` v aplikaci bez podprocesů a ke knihovnám `libmqmxa_r.sl` (nebo `libmqmxa64_r.sl`) a `libmqz_r.sl` v aplikaci s podprocesy.
3. Knihovny produktu WebSphere MQ je třeba propojit před všemi ostatními knihovnami produktu.

Příprava programů v jazyce COBOL v systému HP-UX

Informace o přípravě programů v jazyce COBOL v systému HP-UXs použitím produktu Micro Focus Server Express s produktem WebSphere MQ na platformě IA64 (IPF) a spouštěnými programy v prostředí klienta WebSphere MQ MQI.

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

Poznámky k uživatelům

1. 32bitové příručky pro kopírování v COBOLu jsou instalovány v následujícím adresáři:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy32
   ```

   a symbolické odkazy jsou vytvořeny v:

   ```
   MQ_INSTALLATION_PATH/inc
   ```

2. 64bitové příručky pro kopírování v jazyce COBOL jsou instalovány v následujícím adresáři:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy64
   ```

3. V následujících příkladech nastavte COBCPY na:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy32
   ```

   pro 32bitové aplikace a:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy64
   ```

   pro 64bitové aplikace.

Proveďte kompilaci programů pomocí kompilátoru Micro Focus. Soubory kopií, které deklarují struktury, jsou v adresáři *MQ_INSTALLATION_PATH*/inc:

```
$ export LIB=MQ_INSTALLATION_PATH/lib:$LIB
$ export COBCPY="<COBCPY>"
```

Kompilace 32bitových programů:

```
$ cob32 -xv amqsput.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb Server for COBOL
$ cob32 -xv amqsput.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb Client for COBOL
$ cob32 -xtv amqsput.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb_r Threaded Server for COBOL
$ cob32 -xtv amqsput.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb_r Threaded Client for COBOL
```

Kompilace 64bitových programů:

```
$ cob64 -xv amqsput.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcb Server for COBOL
$ cob64 -xv amqsput.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicb Client for COBOL
$ cob64 -xtv amqsput.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcb_r Threaded Server for COBOL
$ cob64 -xtv amqsput.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicb_r Threaded Client for COBOL
```

kde `amqsput` je ukázkový program.

Ujistěte se, že jste zadali odpovídající velikost zásobníku běhového prostředí; 16 kB je doporučené minimum.

Je třeba propojit své programy s příslušnou knihovnou poskytovanou produktem WebSphere MQ. Následující tabulka ukazuje, která knihovna se má použít v různých prostředích

Hardwarová platforma	Typ programu/ukončení	Soubor knihovny
IA64 (IPF)	Server pro COBOL	libmqmcb.so
IA64 (IPF)	Klient pro COBOL	libmqicb.so
IA64 (IPF)	Aplikace s podprocesy	libmqmcb_r.so

Použití produktu Micro Focus Server Express s produktem WebSphere MQ na platformě IA64 (IPF)

Podrobné informace o používání produktu Micro Focus Server Express ve spojení s produktem WebSphere MQ na platformě HP/IPF naleznete v příručce “Modely adresního prostoru podporované produktem WebSphere MQ for HP-UX on IA64 (IPF)” na stránce 429 .

Programy, které mají být spuštěny v prostředí klienta WebSphere MQ MQI

Pokud používáte pro připojení klienta MQI k serveru logickou jednotku 6.2 , propojte svou aplikaci s produktem libsna.a, který je součástí produktu SNAplusAPI . Použijte volby `-lV3` a `-lstr` u příkazu kompilace a linkování.

- Volba `-lV3` dává vašemu programu přístup k signální knihovně AT & T (SNAplusAPI používá AT & T signály)
- Volba `-lstr` propojí váš program s komponentou proudů.

Příprava programů CICS v systému HP-UX

Naučte se sestavovat transakční programy CICS v systému HP-UX.

Chcete-li sestavit vzorovou transakci CICS , amqscic0.ccs, spusťte následující příkaz:

```
$ export USERLIB="-lmqm_r"
$ cicstcl -l C amqscic0.ccs
```

Je k dispozici modul přepínače XA, který umožňuje propojit produkt CICS s produktem WebSphere MQ:

Tabulka 66. Základní kód pro aplikace CICS (HP-UX)		
Popis	**C (zdroj)**	**C (exec)**
inicializační rutina XA	amqzscix.c	amqzsc

Další informace o podpoře transakcí CICS v produktu Administracenaleznete.

Podpora TXSeries CICS

Produkt WebSphere MQ v systému HP-UX podporuje rozhraní TXSeries CICS pomocí rozhraní XA. Ujistěte se, že aplikace CICS jsou propojeny se vláknovou verzí knihoven MQ .

Zápis programů WebSphere MQ , které jsou načteny do stejné oblasti CICS v jazyce C nebo COBOL. Nemůžete vytvořit kombinaci volání C a COBOL MQI do stejné oblasti CICS . Většina volání MQI ve druhém použitém jazyce se nezdařila s kódem příčiny MQRC_HOBJ_ERROR.

Ukázková transakce CICS C

Ukázkový zdroj C pro transakci CICS WebSphere MQ je poskytnut AMQSCIC0.CCS. Transakce čte zprávy z přenosové fronty SYSTEM.SAMPLE.CICS.WORKQUEUE na výchozím správci front a umisťuje je do lokální fronty s názvem fronty, který je obsažen v záhlaví přenosu zprávy. Veškerá selhání se odesílají do fronty SYSTEM.SAMPLE.CICS.DLQ. Použijte ukázkový skript MQSC AMQSCIC0.TST pro vytvoření těchto front a ukázkových vstupních front.

Příprava programů CICS COBOL pomocí Micro Focus COBOL

Produkt *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, v němž je nainstalován produkt WebSphere MQ .

Chcete-li použít funkci Micro Focus COBOL, postupujte takto:

1. Přidejte modul běhové knihovny produktu WebSphere MQ COBOL do knihovny běhového prostředí pomocí následujícího příkazu:

   ```
   cicsmkcobol -L/usr/lib/dce -LMQ_INSTALLATION_PATH/lib  \
               MQ_INSTALLATION_PATH/lib/libmqmcbrt.o -lmqe_r
   ```

   Poznámka: V produktu `cicsmkcobol`produkt WebSphere MQ neumožňuje provádět volání MQI v programovacím jazyce C z vaší aplikace v jazyku COBOL.

   Pokud mají existující aplikace taková volání, doporučuje se tyto funkce přesunout z aplikací v jazyce COBOL do své vlastní knihovny, například `myMQ.so`. Po přesunu těchto funkcí nezahrnujte knihovnu WebSphere MQ `libmqmcbrt.o` při sestavování aplikace COBOL pro CICS.

   Navíc, pokud vaše aplikace v jazyku COBOL nevolá žádné volání COBOL MQI, nepropojujte `libmqmz_r` s `cicsmkcobol`.

   Tím se vytvoří soubor metody jazyka COBOL Micro Focus a umožní knihovně CICS runtime COBOL zavolat produkt WebSphere MQ na systémy UNIX and Linux .

   Poznámka: Spusťte produkt `cicsmkcobol` pouze při instalaci jednoho z následujících produktů:

   - Nová verze nebo vydání Micro Focus COBOL
   - Nová verze nebo vydání CICS pro HP-UX
   - Nová verze nebo vydání všech podporovaných databázových produktů (pouze pro transakce v jazyce COBOL)
   - Nová verze nebo vydání produktu WebSphere MQ
2. Exportovat následující proměnnou prostředí:

   ```
   COBCPY=MQ_INSTALLATION_PATH/inc  export COBCPY
   ```
3. Přeložte, zkompilujte a propojte program zadáním příkazu:

   ```
   cicstcl -l COBOL -e <yourprog>.ccp
   ```

Modely adresního prostoru podporované produktem WebSphere MQ for HP-UX on IA64 (IPF)

Systém HP-UX nabízí několik modelů adresního prostoru, které mohou využívat aplikace WebSphere MQ .

Systém HP-UX podporuje dva modely adresního prostoru:

- MGAS-Většinou globální adresní prostor (toto je výchozí nastavení a je používán produktem WebSphere MQ)
- MPAS-převážně soukromý adresní prostor

Aplikace, které se připojují k produktu WebSphere MQ , mohou používat modely s adresní prostorem MGAS nebo MPAS. Aplikace vytvořené pomocí modelu MPAS, které se připojují k produktu WebSphere MQ pomocí sdílené paměti, mohou způsobit menší náklady na výkon kvůli neefektivitě při mapování stránek sdílené paměti používaných produktem WebSphere MQ do virtuálního adresního prostoru programu MPAS.

Aplikace v COBOLu sestavené pomocí standardně Micro Focus Server Express používají model MPAS jako model MPAS.

K ověření a změně modelu adresování používaného programem můžete použít program **chatr** .

Pokud se setkáte s problémy s připojením k produktu WebSphere MQ z 32bitových programů MPAS, zvažte použití modelu adresování MGAS nebo sestavení vaší aplikace jako 64bitové aplikace MPAS, spíše než 32bitového aplikace MPAS.

Další podrobnosti o modelech s adresami MGAS a MPAS najdete v dokumentaci k systému HP-UX .

Vytváření vaší aplikace v systému Linux

Tyto informace popisují další úlohy a změny standardních úloh, které musíte provést při sestavování produktu WebSphere MQ pro aplikace Linux ke spuštění.

Jsou podporovány jazyky C a C++. Informace o přípravě programů C + +, viz Použití C++.

Příprava programů jazyka C v produktu Linux

Předkompilované C programy jsou dodávány v adresáři *MQ_INSTALLATION_PATH*/samp/bin . Chcete-li sestavit ukázku ze zdrojového kódu, použijte kompilátor gcc .

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

Pracujte ve svém normálním prostředí. Další informace o programování 64bitových aplikací najdete v tématu Koding standardů na 64bitových platformách.

Propojování knihoven

V následujících tabulkách jsou uvedeny knihovny, které jsou zapotřebí při přípravě programů v jazyce C v systému Linux.

- Je třeba propojit své programy s příslušnou knihovnou poskytovanou produktem WebSphere MQ.

  V prostředí bez podprocesů se připojte k jedné z následujících knihoven:

Soubor knihovny	Typ programu/ukončení
libmqm.so	Server pro C
libmqic.so & libmqm.so	Klient pro C

  V prostředí s podprocesy se připojte k jedné 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. Pokud zapisujete instalovatelnou službu (viz Administrace pro další informace), musíte se připojit ke knihovně `libmqmzf.so` .
2. Pokud vytváříte aplikaci pro externí koordinaci správce transakcí kompatibilní se standardem XA, jako např. IBM TXSeries Encinanebo BEA Tuxedo, musíte se připojit k serveru `libmqmxa.so` (nebo `libmqmxa64.so` , pokud váš správce transakcí zachází s typem " long ' typu 64 bitů) a `libmqz.so` v aplikaci bez podprocesů a ke knihovnám `libmqmxa_r.so` (nebo `libmqmxa64_r.so`) a `libmqz_r.so` v aplikaci s podprocesy.
3. Knihovny produktu WebSphere MQ je třeba propojit před všemi ostatními knihovnami produktu.

Sestavení 31bitových aplikací

Toto téma obsahuje příklady příkazů používaných k sestavení 31bitových programů v různých prostředích.

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

Klientská aplikace C, 31-bitů, bez podprocesů

```
gcc -m31  -o famqsputc_32 amqsput0.c -IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqic
```

Klientská aplikace C, 31bitový, s podporou podprocesů

```
gcc -m31  -o amqsputc_32_r amqsput0.c -IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqic_r -lpthread
```

Serverová aplikace C, 31bitový, bez podprocesů

```
gcc -m31  -o amqsput_32 amqsput0.c -IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqm
```

Serverová aplikace C, 31bitový, s podporou podprocesů

```
gcc -m31  -o amqsput_32_r amqsput0.c -IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqm_r -lpthread
```

Klientská aplikace C + +, 31bitový, bez podprocesů

```
g++ -m31 -fsigned-char  -o imqsputc_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqc23gl
-limqb23gl -lmqic
```

klientská aplikace C + +, 31bitový, s podporou podprocesů

```
g++ -m31 -fsigned-char  -o imqsputc_32_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqc23gl_r
-limqb23gl_r -lmqic_r -lpthread
```

Serverová aplikace C + +, 31 bitů, bez podprocesů.

```
g++ -m31 -fsigned-char  -o imqsput_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqs23gl
-limqb23gl -lmqm
```

Serverová aplikace C + +, 31 bitů, s podprocesy

```
g++ -m31 -fsigned-char  -o imqsput_32_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
```

```
-limqs23gl_r
-limqb23gl_r -lmqm_r -lpthread
```

C client exit, 31-bit, non-threaded

```
gcc -m31 -shared -fPIC  -o /var/mqm/exits/cliexit_32 cliexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib
-Wl,-rpath=/usr/lib
-lmqic
```

C client exit, 31-bit, threaded

```
gcc -m31 -shared -fPIC  -o /var/mqm/exits/cliexit_32_r cliexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib
-Wl,-rpath=/usr/lib
-lmqic_r -lpthread
```

Ukončení serveru C, 31bitový, bez podprocesů.

```
gcc -m31 -shared -fPIC  -o /var/mqm/exits/srvexit_32 srvexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib
-Wl,-rpath=/usr/lib
-lmqm
```

Ukončení serveru C, 31bitový, s podporou podprocesů

```
gcc -m31 -shared -fPIC  -o /var/mqm/exits/srvexit_32_r srvexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib
-Wl,-rpath=/usr/lib
-lmqm_r -lpthread
```

Sestavování 32 bitových aplikací

Toto téma obsahuje příklady příkazů použitých pro sestavení 32bitových programů v různých prostředích.

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

Klientská aplikace C, 32bitová, bez podprocesů

```
gcc -m32 -o amqsputc_32 amqsput0.c -IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqic
```

Klientská aplikace C, 32bitová, se závitem

```
gcc -m32 -o amqsputc_32_r amqsput0.c -IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqic_r -lpthread
```

Serverová aplikace C, 32bitová, bez podprocesů

```
gcc -m32 -o amqsput_32 amqsput0.c -IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqm
```

Serverová aplikace C, 32bitová, se závitem

```
gcc -m32 -o amqsput_32_r amqsput0.c -IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqm_r -lpthread
```

Klientská aplikace C + +, 32bitová, bez podprocesů

```
g++ -m32 -fsigned-char -o imqsputc_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
```

```
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib  -Wl,-rpath=/usr/lib
-limqc23gl -limqb23gl -lmqic
```

Klientská aplikace C + +, 32bitová verze

```
g++ -m32 -fsigned-char -o imqsputc_32_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib  -Wl,-rpath=/usr/lib
-limqc23gl_r -limqb23gl_r -lmqic_r -lpthread
```

Serverová aplikace C + +, 32bitová, bez podprocesů

```
g++ -m32 -fsigned-char -o imqsput_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqs23gl -limqb23gl -lmqm
```

Serverová aplikace C + +, 32bitová verze

```
g++ -m32 -fsigned-char -o imqsput_32_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqs23gl_r -limqb23gl_r -lmqm_r -lpthread
```

C client exit, 32bitový, non-threaded

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/cliexit_32 cliexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib
-Wl,-rpath=/usr/lib
-lmqic
```

C client exit, 32bitový, threaded

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/cliexit_32_r cliexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib
-Wl,-rpath=/usr/lib
-lmqic_r -lpthread
```

C server exit, 32bitový, bez podprocesů

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/srvexit_32 srvexit.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqm
```

C server exit, 32bitový, threaded

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/srvexit_32_r srvexit.c
IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib
-Wl,-rpath=/usr/lib
lmqm_r -lpthread
```

Sestavování 64bitových aplikací

Toto téma obsahuje příklady příkazů používaných k sestavení 64bitových programů v různých prostředích.

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

Klientská aplikace C, 64bitová, bez podprocesů

```
gcc -m64  -o amqsputc_64 amqsput0.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -lmqic
```

C client application, 64-bit, threaded

```
gcc -m64  -o amqsputc_64_r amqsput0.c -IMQ_INSTALLATION_PATH/inc
```

```
-LMQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -lmqic_r
-lpthread
```

Serverová aplikace C, 64bitová, bez podprocesů

```
gcc -m64  -o amqsput_64 amqsput0.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -lmqm
```

C server application, 64-bit, threaded

```
gcc -m64  -o amqsput_64_r amqsput0.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -lmqm_r
-lpthread
```

Klientská aplikace C + +, 64bitová, bez podprocesů

```
g++ -m64 -fsigned-char -o imqsputc_64 imqsput.cpp
-IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqc23gl -limqb23gl -lmqic
```

Klientská aplikace C + +, 64bitová verze

```
g++ -m64 -fsigned-char -o imqsputc_64_r imqsput.cpp
-IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqc23gl_r -limqb23gl_r -lmqic_r -lpthread
```

Serverová aplikace C + +, 64bitová, bez podprocesů

```
g++ -m64 -fsigned-char  -o imqsput_64 imqsput.cpp
-IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=/usr/lib64 -limqs23gl -limqb23gl -lmqm
```

Serverová aplikace C + +, 64bitová verze

```
g++ -m64 -fsigned-char  -o imqsput_64_r imqsput.cpp
-IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=/usr/lib64 -limqs23gl_r -limqb23gl_r -lmqm_r -lpthread
```

C exit exit, 64-bit, bez podprocesů

```
gcc -m64 -shared -fPIC  -o /var/mqm/exits64/cliexit_64 cliexit.c
-IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=/usr/lib64 -lmqic
```

C exit exit, 64-bit, threaded

```
gcc -m64 -shared -fPIC  -o /var/mqm/exits64/cliexit_64_r cliexit.c
-IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=/usr/lib64 -lmqic_r -lpthread
```

C server exit, 64-bit, bez podprocesů

```
gcc -m64 -shared -fPIC  -o /var/mqm/exits64/srvexit_64 srvexit.c
-IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=/usr/lib64 -lmqm
```

C server exit, 64-bit, threaded

```
gcc -m64 -shared -fPIC  -o /var/mqm/exits64/srvexit_64_r srvexit.c
-IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=/usr/lib64 -lmqm_r -lpthread
```

Příprava programů COBOL v produktu Linux

Informace o přípravě programů v jazyce COBOL v produktu Linux a přípravě programů v jazyce COBOL pomocí funkce Micro Focus COBOL.

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

1. 32bitové příručky pro kopírování v COBOLu jsou instalovány v následujícím adresáři:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy32
   ```

   a symbolické odkazy jsou vytvořeny v:

   ```
   MQ_INSTALLATION_PATH/inc
   ```

2. Na 64bitových platformách jsou 64bitové knihy pro kopírování COBOL instalovány v následujícím adresáři:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy64
   ```

3. V následujících příkladech nastavte COBCPY na:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy32
   ```

   pro 32bitové aplikace a:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy64
   ```

   pro 64bitové aplikace.

Musíte propojit svůj program s jednou z následujících možností:

Soubor knihovny	Typ programu/ukončení
libmqmcb.so	Server pro COBOL
libmqicb.so	Klient pro COBOL
libmqmcb_r.so	Server pro COBOL (podprocesová aplikace)
libmqicb_r.so	Klient pro COBOL (podprocesová aplikace)

Příprava programů v jazyce COBOL pomocí Micro Focus COBOL

Před kompilací programu nastavte proměnné prostředí následujícím způsobem:

```
export COBCPY=<COBCPY>
export LIB=MQ_INSTALLATION_PATHlib:$LIB
```

Chcete-li zkompilovat 32bitový program COBOL, je-li podporován, použijte Micro Focus COBOL, zadejte:

```
$ cob32 -xvP amqsput.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb Server for COBOL
$ cob32 -xvP amqsput.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb Client for COBOL
$ cob32 -xtvP amqsput.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb_r Threaded Server for COBOL
$ cob32 -xtvP amqsput.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb_r Threaded Client for COBOL
```

Chcete-li zkompilovat 64bitový program COBOL pomocí Micro Focus COBOL, zadejte:

```
$ cob64 -xvP amqsput.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcb Server for COBOL
$ cob64 -xvP amqsput.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicb Client for COBOL
$ cob64 -xtvP amqsput.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcb_r Threaded Server for COBOL
$ cob64 -xtvP amqsput.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicb_r Threaded Client for COBOL
```

kde `amqsput` je ukázkový program.

Popis proměnných prostředí, které potřebujete, najdete v dokumentaci Micro Focus COBOL.

Vytváření vaší aplikace v systému Solaris

Tyto informace popisují další úlohy a změny standardních úloh, které musíte provést při sestavování produktu WebSphere MQ pro aplikace systému Solaris, které se mají spustit v systému Solaris.

Jsou podporovány programovací jazyky COBOL, C a C++. Informace o přípravě programů C + +, viz Použití C++.

Kromě kódování volání MQI ve vašem zdrojovém kódu je třeba přidat příslušné soubory začlenění. Seznamte se s obsahem těchto souborů. Úplný popis najdete v tématu “Soubory definic dat produktu IBM WebSphere MQ” na stránce 77 .

V celém tomto tématu se znak zpětného lomítka (\) používá k rozdělení dlouhých příkazů na více než jeden řádek. Nezadávejte tento znak, zadejte každý příkaz jako jednu řádku.

Příprava programů jazyka C v systému Solaris

Předkompilované C programy jsou dodávány v adresáři *MQ_INSTALLATION_PATH*`/samp/bin` .

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

Další informace o programování 64bitových aplikací najdete v tématu Koding standardů na 64bitových platformách.

Chcete-li používat programy na počítači, který má nainstalován pouze klient WebSphere MQ MQI pro Solaris, zkompilujte programy tak, aby je propojovaly s knihovnou klienta (`-lmqic`).

Pokud použijete nepodporovaný kompilátor `?usr?ucb?cc`, může se vaše aplikace zkompilovat a úspěšně propojit. Když však spustíte aplikaci, nezdaří se, když se pokusí připojit ke správci front.

Poznámka: 32bitoví klienti Solaris x86 s protokolem SSL a TLS, nakonfigurovaní pro operace dle standardu FIPS 140-2, selžou, budete-li je provozovat na systémech Intel. K tomuto selhání dochází, protože soubor 32bitové knihovny GSKit-Crypto Solaris x86 vyhovující specifikaci FIPS 140-2 se do čipové sady Intel nenačte. V zasažených systémech se do protokolu chyb klienta nahlásí chyba AMQ9655. Tento problém vyřešíte tak, že vypnete kompatibilitu se specifikací FIPS 140-2, nebo znovu zkompilujete aplikaci klienta pro 64 bitů, protože 64bitový kód není dotčen.

Propojování knihoven

Musíte propojit s knihovnami produktu WebSphere MQ , které jsou vhodné pro daný typ aplikace:

Soubory knihovny	Typ programu/ukončení
libmqm.so	Server pro C
libmqic.so & libmqm.so	Klient pro C

Poznámka:

1. Pokud píšete instalovatelnou službu (další informace viz Administrace), propojte se s knihovnou `libmqmzf.so` .
2. Pokud vytváříte aplikaci pro externí koordinaci správce transakcí kompatibilní se standardem XA, jako např. IBM TXSeries Encinanebo BEA Tuxedo, musíte se připojit k serveru `libmqmxa.so` (nebo `libmqmxa64.so` , pokud váš správce transakcí zachází s typem " long ' typu 64 bit) a s knihovnami `libmqz.so` .
3. Knihovny produktu WebSphere MQ je třeba propojit před všemi ostatními knihovnami produktu.

Sestavování aplikací v systému x86-64

Toto téma obsahuje příklady příkazů používaných k sestavování programů v různých prostředích na platformě x86-64 .

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

Klientská aplikace C, 32bitová

```
cc -xarch=386 -mt -o amqsputc_32 amqsput0.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib
-RMQ_INSTALLATION_PATH/lib -R/usr/lib/32 -lmqic -lsocket -lnsl -ldl
```

Klientská aplikace C, 64bitová

```
cc -xarch=amd64 -mt -o amqsputc_64 amqsput0.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -lmqic -lsocket
-lnsl -ldl
```

Serverová aplikace C, 32 bitů

```
cc -xarch=386 -mt -o amqsput_32 amqsput0.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib
-RMQ_INSTALLATION_PATH/lib -R/usr/lib/32 -lmqm -lsocket -lnsl -ldl
```

Serverová aplikace C, 64bitová

```
cc -xarch=amd64 -mt -o amqsput_64 amqsput0.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -lmqm -lsocket
-lnsl -ldl
```

C++ client application, 32-bit

```
CC -xarch=386 -mt -o imqsputc_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib
-RMQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqc23as -limqb23as -lmqic -lsocket -lnsl -ldl
```

Klientská aplikace C + +, 64bitová

```
CC -xarch=amd64 -mt -o imqsputc_64 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqc23as
```

```
-limqb23as
-lmqic -lsocket -lnsl -ldl
```

Serverová aplikace C + +, 32bitová

```
CC -xarch=386 -mt -o imqsput_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib
-RMQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqs23as -limqb23as -lmqm
-lsocket -lnsl -ldl
```

Serverová aplikace C + +, 64bitový

```
CC -xarch=amd64 -mt -o imqsput_64 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqs23as
-limqb23as -lmqm
-lsocket -lnsl -ldl
```

C client exit, 32-bit

```
cc -xarch=386 -mt -G -KPIC -o /var/mqm/exits/cliexit_32 cliexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -RMQ_INSTALLATION_PATH/lib
-R/usr/lib/32 -lmqic
-lsocket -lnsl -ldl
```

C client exit, 64-bit

```
cc -xarch=amd64 -mt -G -KPIC -o /var/mqm/exits64/cliexit_64 cliexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64
-R/usr/lib/64 -lmqic
-lsocket -lnsl -ldl
```

Ukončení serveru C, 32 bitů

```
cc -xarch=386 -mt -G -KPIC -o /var/mqm/exits/srvexit_32 srvexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -RMQ_INSTALLATION_PATH/lib
-R/usr/lib/32 -lmqm
-lsocket -lnsl -ldl
```

Ukončení serveru C, 64bitový

```
cc -xarch=amd64 -mt -G -KPIC -o /var/mqm/exits64/srvexit_64 srvexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64
-R/usr/lib/64 -lmqm
-lsocket -lnsl -ldl
```

Sestavování aplikací na platformě SPARC

Toto téma obsahuje příklady příkazů používaných k sestavování programů v různých prostředích na platformě SPARC.

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

Klientská aplikace C, 32bitová

```
cc -xarch=v8plus -mt -o amqsputc_32 amqsput0.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib
-RMQ_INSTALLATION_PATH/lib -R/usr/lib/32 -lmqic -lsocket -lnsl -ldl
```

Klientská aplikace C, 64bitová

```
cc -xarch=v9 -mt -o amqsputc_64 amqsput0.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -lmqic
-lsocket -lnsl -ldl
```

Serverová aplikace C, 32 bitů

```
cc -xarch=v8plus -mt -o amqsput_32 amqsput0.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib
-RMQ_INSTALLATION_PATH/lib -R/usr/lib/32 -lmqm -lsocket -lnsl -ldl
```

Serverová aplikace C, 64bitová

```
cc -xarch=v9 -mt -o amqsput_64 amqsput0.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -lmqm
-lsocket -lnsl -ldl
```

C++ client application, 32-bit

```
CC -xarch=v8plus -mt -o imqsputc_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -RMQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqc23as -limqb23as
-lmqic
-lsocket -lnsl -ldl
```

Klientská aplikace C + +, 64bitová

```
CC -xarch=v9 -mt -o imqsputc_64 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqc23as
-limqb23as
-lmqic -lsocket -lnsl -ldl
```

Serverová aplikace C + +, 32bitová

```
CC -xarch=v8plus -mt -o imqsput_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib
-RMQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqs23as -limqb23as -lmqm
-lsocket -lnsl -ldl
```

Serverová aplikace C + +, 64bitový

```
CC -xarch=v9 -mt -o imqsput_64 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqs23as
-limqb23as -lmqm
-lsocket -lnsl -ldl
```

C client exit, 32-bit

```
cc -xarch=v8plus -mt -G -KPIC -o /var/mqm/exits/cliexit_32 cliexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -RMQ_INSTALLATION_PATH/lib
-R/usr/lib/32 -lmqic
-lsocket -lnsl -ldl
```

C client exit, 64-bit

```
cc -xarch=v9 -mt -G -KPIC -o /var/mqm/exits64/cliexit_64 cliexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64
-R/usr/lib/64 -lmqic
-lsocket -lnsl -ldl
```

Ukončení serveru C, 32 bitů

```
cc -xarch=v8plus -mt -G -KPIC -o /var/mqm/exits/srvexit_32 srvexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -RMQ_INSTALLATION_PATH/lib
-R/usr/lib/32 -lmqm
-lsocket -lnsl -ldl
```

Ukončení serveru C, 64bitový

```
cc -xarch=v9 -mt -G -KPIC -o /var/mqm/exits64/srvexit_64 srvexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64
-R/usr/lib/64 -lmqm
-lsocket -lnsl -ldl
```

Příprava programů COBOL v systému Solaris

Informace o přípravě programů v jazyce COBOL v systému Solaris.

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

1. 32bitové příručky pro kopírování v COBOLu jsou instalovány v následujícím adresáři:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy32
   ```

   a symbolické odkazy jsou vytvořeny v:

   ```
   MQ_INSTALLATION_PATH/inc
   ```

2. 64bitové příručky pro kopírování v jazyce COBOL jsou instalovány v následujícím adresáři:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy64
   ```

3. V následujících příkladech nastavte COBCPY na:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy32
   ```

   pro 32bitové aplikace a:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy64
   ```

   pro 64bitové aplikace.

Proveďte kompilaci programů pomocí kompilátoru Micro Focus. Kopírovaté soubory, které deklarují struktury, jsou v produktu *MQ_INSTALLATION_PATH*/inc:

```
$ export LIB=MQ_INSTALLATION_PATH/lib:$LIB
$ export COBCPY="<COBCPY>"
```

Kompilace 32bitových programů:

- `$ cob32 -xv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb`

  Server pro COBOL
- `$ cob32 -xv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb`

  Klient pro COBOL
- `$ cob32 -xtv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb_r`

  Threadovaný server pro COBOL
- `$ cob32 -xtv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb_r`

  Klient s podporou podprocesů pro COBOL

Kompilace 64-bitových programů:

- `$ cob64 -xv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcb`

  Server pro COBOL

- `$ cob64 -xv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicb`

  Klient pro COBOL
- `$ cob64 -xtv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcb_r`

  Threadovaný server pro COBOL
- `$ cob64 -xtv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicb_r`

  Klient s podporou podprocesů pro COBOL

kde *amqs0put0.cbl* je ukázkový program.

Musíte propojit svůj program s jednou z následujících možností:

- libmqmcb.so

  Server pro COBOL
- libmqicb.so

  Klient pro COBOL

Příprava programů CICS v systému Solaris

Informace o přípravě programů CICS v systému Solaris.

Je k dispozici modul přepínače XA, který umožňuje propojit produkt CICS s produktem WebSphere MQ:

Tabulka 67. Základní kód pro aplikace CICS (Solaris)		
Popis	**C (zdroj)**	**C (exec)**
inicializační rutina XA	amqzscix.c	amqzsc- TXSeries pro Solaris

Vždy propojte své transakce s bezpečným produktem WebSphere MQ knihovnou libmqm.so.

Další informace o podpoře transakcí CICS v produktu Administracenaleznete.

Podpora TXSeries CICS

Produkt WebSphere MQ for Solaris podporuje rozhraní TXSeries CICS pomocí rozhraní XA.

Zápis programů WebSphere MQ , které jsou načteny do stejné oblasti CICS v jazyce C nebo COBOL. Nemůžete vytvořit kombinaci volání C a COBOL MQI do stejné oblasti CICS . Většina volání MQI ve druhém použitém jazyce se nezdařila s kódem příčiny `MQRC_HOBJ_ERROR`.

Příprava programů CICS COBOL pomocí Micro Focus COBOL

Produkt *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

Chcete-li použít funkci Micro Focus COBOL, postupujte takto:

1. Přidejte modul běhové knihovny produktu WebSphere MQ COBOL do knihovny běhového prostředí pomocí následujícího příkazu:

   ```
   cicsmkcobol -L/usr/lib/dce -LMQ_INSTALLATION_PATH/lib  \
               MQ_INSTALLATION_PATH/lib/libmqmcbrt.o -lmqe
   ```

   Poznámka: V produktu `cicsmkcobol`produkt WebSphere MQ neumožňuje provádět volání MQI v programovacím jazyce C z vaší aplikace v jazyku COBOL.

Mají-li existující aplikace taková volání, přesuňte tyto funkce z aplikací v jazyce COBOL do vaší vlastní knihovny, například `myMQ.so`. Po přesunu těchto funkcí nezahrnujte knihovnu WebSphere MQ `libmqmcbrt.o` při sestavování aplikace COBOL pro CICS.

Navíc, pokud vaše aplikace v jazyku COBOL nevolá žádné volání COBOL MQI, nepropojujte `libmqmz_r` s `cicsmkcobol`.

Tím se vytvoří soubor metody jazyka COBOL Micro Focus a umožní knihovně CICS runtime COBOL zavolat produkt WebSphere MQ na systémy UNIX and Linux .

Poznámka: Spusťte produkt `cicsmkcobol` pouze při instalaci jednoho z následujících produktů:

- Nová verze nebo vydání Micro Focus COBOL
- Nová verze nebo vydání TXSeries pro Solaris
- Nová verze nebo vydání všech podporovaných databázových produktů (pouze pro transakce v jazyce COBOL)
- Nová verze nebo vydání produktu WebSphere MQ

2. Exportovat následující proměnnou prostředí:

```
COBCPY=MQ_INSTALLATION_PATH/inc  export COBCPY
```

3. Přeložte, zkompilujte a propojte program zadáním příkazu:

```
cicstcl -l COBOL -e <yourprog>.ccp
```

Příprava programů CICS C

Sestavte programy CICS C pomocí standardních zařízení CICS :

1. Exportujte ***jeden*** z následujících proměnných prostředí:
   - LDFLAGS = "-L*MQ_INSTALLATION_PATH*l lib -lmqm_r" export LDFLAGS
   - USERLIB = "-L*MQ_INSTALLATION_PATH*l lib -lmqm_r" export USERLIB
2. Přeložte, zkompilujte a propojte program zadáním příkazu:

```
cicstcl -l C amqscic0.ccs
```

Ukázková transakce CICS C

Ukázkový zdroj C pro transakci CICS WebSphere MQ je poskytnut AMQSCIC0.CCS. Transakce čte zprávy z přenosové fronty SYSTEM.SAMPLE.CICS.WORKQUEUE na výchozím správci front a umístěte je do lokální fronty s názvem fronty, který je obsažen v záhlaví přenosu zprávy. Veškerá selhání se odesílají do fronty SYSTEM.SAMPLE.CICS.DLQ. Použijte ukázkový skript MQSC AMQSCIC0.TST pro vytvoření těchto front a ukázkových vstupních front.

Vytváření vaší aplikace v systémech Windows

Příručky systému Windows popisují, jak vytvářet spustitelné aplikace z programů, které napíšete.

Toto téma popisuje další úlohy a změny standardních úloh, které musíte provést při sestavování aplikací WebSphere MQ for Windows pro spuštění v systémech Windows . Jsou podporovány programovací jazyky ActiveX, C, C + +, COBOL a Visual Basic. Informace o přípravě programů ActiveX naleznete v části Použití rozhraní modelu COM (Component Object Model) (WebSphere MQ Automation Classes for ActiveX). Informace o přípravě programů C + +, viz Použití C++.

Úlohy, které je třeba provést při vytváření spustitelné aplikace pomocí produktu WebSphere MQ for Windows , se liší podle programovacího jazyka, ve kterém je napsán váš zdrojový kód. Kromě kódování volání MQI ve vašem zdrojovém kódu musíte přidat příslušné příkazy jazyka pro zahrnutí produktu

WebSphere MQ for Windows do souborů začlenění pro jazyk, který používáte. Seznamte se s obsahem těchto souborů. Úplný popis najdete v tématu “Soubory definic dat produktu IBM WebSphere MQ” na stránce 77 .

Sestavování 64bitových aplikací v systému Windows

Oba 32bitové i 64bitové aplikace jsou podporovány v systému IBM WebSphere MQ for Windows Version 7.5. Spustitelné soubory a soubory knihovny produktu IBM WebSphere MQ jsou dodávány v 32bitové i 64bitové formě, použijte příslušnou verzi v závislosti na aplikaci, se kterou pracujete.

Spustitelné soubory a knihovny

32bitové i 64bitové verze knihoven produktu IBM WebSphere MQ jsou dodávány v následujících umístěních:

Tabulka 68. Umístění knihoven IBM WebSphere 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, do kterého je produkt WebSphere MQ nainstalován.

32bitové aplikace pokračují v práci normálně po migraci. 32 bitové soubory existují ve stejném adresáři jako v předchozích verzích produktu.

Chcete-li vytvořit 64bitovou verzi, musíte se ujistit, že je vaše prostředí nakonfigurováno pro použití souborů knihovny v produktu *MQ_INSTALLATION_PATH*\Tools\Lib64. Ujistěte se, že proměnná prostředí LIB není nastavena, aby se podíval do složky obsahující 32bitové knihovny.

Příprava programů jazyka C v systému Windows

Pracujte v typickém prostředí Windows ; produkt WebSphere MQ for Windows nevyžaduje nic speciálního.

Další informace o programování 64bitových aplikací najdete v tématu Kodové standardy na 64bitových platformách.

- Propojte své programy s odpovídajícími knihovnami poskytnutými produktem WebSphere MQ:

Soubor knihovny	**Typ programu/ukončení**
MQ_INSTALLATION_PATH\Tools\Lib\mqm.lib	server pro 32bitovou verzi C
MQ_INSTALLATION_PATH\Tools\Lib\mqic.lib	klient pro 32bitovou verzi C
MQ_INSTALLATION_PATH\Tools\Lib\mqicxa.lib	klient pro 32bitovou transakci s koordinačním transakcí
MQ_INSTALLATION_PATH\Tools\Lib64\mqm.lib	server pro 64 bitů C

Soubor knihovny	Typ programu/ukončení
MQ_INSTALLATION_PATH\Tools\Lib64\mqic.lib	klient pro 64 bitů C
MQ_INSTALLATION_PATH\Tools\Lib64\mqicxa.lib	klient pro 64bitový C s koordinací transakcí

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

Následující příkaz dává příklad kompilace ukázkového programu amqsget0 (pomocí kompilátoru Microsoft Visual C + +).

Pro 32 bitové aplikace:

```
cl -MD amqsget0.c -Feamqsget.exe MQ_INSTALLATION_PATH\Tools\Lib\mqm.lib
```

Pro 64bitové aplikace:

```
cl -MD amqsget0.c -Feamqsget.exe MQ_INSTALLATION_PATH\Tools\Lib64\mqm.lib
```

Poznámka:

- Pokud píšete instalovatelnou službu (viz Administrace pro další informace), musíte se odkázat na knihovnu mqmzf.lib .
- Pokud vytváříte aplikaci pro externí koordinaci správce transakcí kompatibilní se standardem XA, jako např. IBM TXSeries Encinanebo BEA Tuxedo, musíte vytvořit odkaz na knihovnu mqmxa.lib nebo mqmxa.lib .
- Pokud vytváříte uživatelskou proceduru CICS , propojte ji s knihovnou mqmcics4.lib .
- Knihovny produktu WebSphere MQ je třeba propojit před všemi ostatními knihovnami produktu.

- Knihovny DLL se musí nacházet v cestě (PATH), kterou jste zadali.
- Pokud použijete malá písmena, je-li to možné, můžete přejít z produktu WebSphere MQ for Windows na produkt WebSphere MQ na systémech UNIX and Linux , kde je třeba použít malá písmena.

Příprava programů CICS a Transaction Server

Ukázkový zdroj C pro transakci CICS WebSphere MQ je poskytnut AMQSCIC0.CCS. Sestavujete ji pomocí standardního zařízení CICS . Například pro TXSeries pro Windows 2000:

1. Nastavte proměnnou prostředí (zadejte na jednom řádku následující kód):

   ```
   set CICS_IBMC_FLAGS=-IMQ_INSTALLATION_PATH\Tools\C\Include;
   %CICS_IBMC_FLAGS%
   ```

2. Nastavte proměnnou prostředí USERLIB:

   ```
   set USERLIB=MQM.LIB;%USERLIB%
   ```

3. Translate, compile, and link the sample program:

   ```
   cicstcl -l IBMC amqscic0.ccs
   ```

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

To je popsáno v příručce *Transaction Server for Windows NT Application Programming Guide (CICS) V4*.

Další informace o podpoře transakcí CICS v produktu Administracenaleznete.

Příprava programů COBOL v produktu Windows

Tyto informace použijte k seznámení se s programy v jazyce COBOL v produktu Windowsa k přípravě programů CICS a transakčních serverů.

1. 32bitové příručky pro kopírování v jazyce COBOL jsou instalovány v následujícím adresáři: *MQ_INSTALLATION_PATH*\Tools\cobol\CopyBook.
2. Příručky pro kopírování 64bitového jazyka COBOL jsou instalovány v následujícím adresáři: *MQ_INSTALLATION_PATH*\Tools\cobol\CopyBook64
3. V následujících příkladech nastavte CopyBook na:

   ```
   CopyBook
   ```

   pro 32bitové aplikace a:

   ```
   CopyBook64
   ```

   pro 64bitové aplikace.

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

Chcete-li připravit programy v jazyce COBOL na systémech Windows , propojte svůj program s jednou z následujících knihoven poskytnutých produktem IBM WebSphere MQ:

Soubor knihovny	Typ programu nebo ukončení
MQ_INSTALLATION_PATH\Tools\Lib\mqmcbb	32bitový server pro IBM COBOL
MQ_INSTALLATION_PATH\Tools\Lib\mqmcb	32bitový server pro Micro COBOL
MQ_INSTALLATION_PATH\Tools\Lib\mqiccbb	32bitový klient pro IBM COBOL
MQ_INSTALLATION_PATH\Tools\Lib\mqiccb	32bitový klient pro Micro COBOL
MQ_INSTALLATION_PATH\Tools\Lib64\mqmcbb	64bitový server pro IBM COBOL
MQ_INSTALLATION_PATH\Tools\Lib64\mqmcb	64bitový server pro Micro COBOL
MQ_INSTALLATION_PATH\Tools\Lib64\mqiccbb	64bitový klient pro IBM COBOL
MQ_INSTALLATION_PATH\Tools\Lib64\mqiccb	64bitový klient pro funkci Micro Focus COBOL

Pokud spouštíte program v prostředí klienta MQI, ujistěte se, že se knihovna DOSCALLS objeví před jakoukoli knihovnou COBOL nebo IBM WebSphere MQ .

V závislosti na programu můžete použít kompilátor IBM COBOL Set nebo Micro Focus COBOL compiler.

- Programy začínající amqi jsou vhodné pro kompilátor jazyka IBM COBOL Set,
- Programy začínající amqm jsou vhodné pro kompilátor Micro Focus COBOL a
- Programy začínající amq0 jsou vhodné pro kompilátor.

IBM a Micro Focus COBOL

Opětovné připojení existujícího 32bitového programu IBM WebSphere MQ Micro Focus COBOL za použití mqmcb.lib nebo mqiccb.lib, spíše než knihovny mqmcbb a mqiccbb .

Chcete-li kompilovat, například ukázkový program amq0put0, pomocí produktu IBM VisualAge COBOL:

1. Nastavte proměnnou prostředí SYSLIB tak, aby obsahovala cestu k zakladači COBOL produktu IBM WebSphere MQ VisualAge (zadejte následující kód na jednom řádku):

```
set SYSLIB=MQ_INSTALLATION_PATH\
Tools\Cobol\Copybook\VAcobol;%SYSLIB%
```

2. Pro použití na serveru IBM WebSphere MQ :

```
cob2 amq0put0.cbl -qlib "MQ_INSTALLATION_PATH\
Tools\Lib\mqmcbb.lib"
```

3. Pro použití na klientovi IBM WebSphere MQ :

```
cob2 amq0put0.cbl -qlib "MQ_INSTALLATION_PATH\
Tools\Lib\mqiccbb.lib"
```

Poznámka: I když musíte použít volbu kompilátoru `CALLINT (SYSTEM)`, toto je výchozí nastavení pro `cob2`.

Chcete-li kompilovat, například ukázkový program `amq0put0`, pomocí Micro Focus COBOL:

1. Nastavte proměnnou prostředí COBCPY tak, aby ukazovala na zakladače COBOL IBM WebSphere MQ (zadejte následující kód na jednom řádku):

```
set COBCPY=MQ_INSTALLATION_PATH\
Tools\Cobol\Copybook
```

2. Kompilujte program, aby vám dal objektový soubor:

```
cobol amq0put0 LITLINK
```

3. Propojte soubor objektu se systémem běhového prostředí.
   - Nastavte proměnnou prostředí LIB tak, aby ukazovala na knihovny COBOL kompilátoru.
   - Propojte soubor objektu pro použití na serveru IBM WebSphere MQ :

     ```
     cbllink amq0put0.obj mqmcb.lib
     ```

   - Nebo propojte soubor s objektem pro použití na klientovi IBM WebSphere MQ :

     ```
     cbllink amq0put0.obj mqiccb.lib
     ```

Příprava programů CICS a transakčních serverů

Chcete-li kompilovat a propojit TXSeries pro produkt Windows NT, program V5.1 pomocí produktu IBM VisualAge COBOL:

1. Nastavte proměnnou prostředí (zadejte na jednom řádku následující kód):

```
set CICS_IBMCOB_FLAGS=MQ_INSTALLATION_PATH\
Cobol\Copybook\VAcobol;%CICS_IBMCOB_FLAGS%
```

2. Nastavte proměnnou prostředí USERLIB:

```
set USERLIB=MQMCBB.LIB
```

3. Translate, compile, and link your program:

```
cicstcl -l IBMCOB myprog.ccp
```

To je popsáno v příručce *Transaction Server for Windows NT, V4 Application Programming* Guide.

Chcete-li kompilovat a propojit CICS pro program Windows V5 pomocí Micro Focus COBOL, postupujte takto:

- Nastavte proměnnou INCLUDE:

```
set
INCLUDE=<drive>:\<programname>\ibm\websphere\tools\c\include;
        <drive>:\opt\cics\include;%INCLUDE%
```

- Nastavte proměnnou prostředí COBCPY:

```
 setCOBCPY=<drive>:\<programname>\ibm\websphere\tools\cobol\copybook;
           <drive>:\opt\cics\include
```

- Nastavte volby jazyka COBOL:
  - `set`
  - `COBOPTS=/LITLINK /NOTRUNC`

a spusťte následující kód:

```
cicstran cicsmq00.ccp
cobol cicsmq00.cbl /LITLINK /NOTRUNC
cbllink -D -Mcicsmq00 -Ocicsmq00.cbmfnt cicsmq00.obj
%CICSLIB%\cicsprCBMFNT.lib user32.lib msvcrt.lib kernel32.lib mqmcb.lib
```

Příprava programů jazyka Visual Basic v systému Windows

Tyto informace použijte při zvažování použití programů jazyka Visual Basic v systému Windows.

Poznámka: 64bitové verze souborů modulu Visual Basic nejsou dodány.

Příprava programů Visual Basic v systému Windows:

1. Vytvořit nový projekt
2. Přidejte dodaný soubor modulu, CMQB.BASdo projektu.
3. Pokud je potřebujete, přidejte další dodané soubory modulu:

CMQBB.BAS	Podpora MQAI
CMQCFB.BAS	Podpora PCF
CMQXB.BAS	Podpora uživatelských procedur kanálu
CMQPSB.BAS	Publikování/odběr

Informace o použití volání MQCONNXAny v rámci Visual Basic naleznete v příručce “Kódování ve Visual Basicu” na stránce 83 .

Před provedením jakýchkoli volání MQI v kódu projektu volejte proceduru MQ_SETDEFAULTS. Tato procedura nastaví výchozí struktury, které vyžaduje volání MQI.

Určete, zda vytváříte server nebo klienta WebSphere MQ , před kompilací nebo spuštěním projektu nastavením proměnné pro podmíněnou kompilaci *MqType*. Nastavte *MqType* v projektu Visual Basic na 1 pro server nebo 2 pro klienta následujícím způsobem:

1. Vyberte nabídku Projekt.
2. Vyberte položku *Name* Vlastnosti (kde *Name* je název aktuálního projektu).

3. Vyberte kartu Make v dialogovém okně.
4. V poli Argumenty podmíněné kompilace zadejte tento typ pro server:

```
MqType=1
```

nebo toto pro klienta:

```
MqType=2
```

Uživatelská procedura zabezpečení SSPI

Produkt WebSphere MQ for Windows poskytuje uživatelskou proceduru pro zabezpečení pro klienta WebSphere MQ MQI i pro server WebSphere MQ . Jedná se o program výstupního bodu kanálu, který poskytuje ověření pro kanály produktu WebSphere MQ pomocí rozhraní SSPI (Security Services Programming Interface). SSPI poskytuje integrovaná bezpečnostní zařízení pro systémy Windows .

Balíky zabezpečení jsou načteny z adresáře security.dll nebo secur32.dll. Tyto knihovny DLL se dodávají spolu s operačním systémem.

Jednosměrné ověření se poskytuje pomocí ověřovacích služeb NTLM. Dvojcestné ověření je poskytnuto pomocí ověřovacích služeb Kerberos .

Uživatelský program zabezpečení je dodáván ve zdrojovém formátu a ve formátu objektu. Kód objektu můžete použít tak, jak je, nebo můžete použít zdrojový kód jako výchozí bod k vytvoření svých vlastních uživatelských programů.

Další informace najdete v tématu “Použití uživatelské procedury zabezpečení SSPI v systémech Windows” na stránce 163.

Úvod do uživatelských procedur pro zabezpečení

Procedura zabezpečení vytvoří zabezpečené připojení mezi dvěma programy procedury zabezpečení. Jeden z těchto programů odpovídá odesílajícímu agentu MCA (message channel agent) a druhý přijímajícímu agentu MCA.

Program, který iniciuje zabezpečené připojení, tj. první program, který získá řízení po zřízení relace MCA, je znám jako *kontextový iniciátor*. Partnerský program je znám jako *služba kontextu kontextu*.

V následující tabulce jsou uvedeny některé typy kanálů, které jsou inicializátory kontextu a jejich přidružené kontextové akceptory.

Tabulka 69. Kontextové iniciátory a jejich přidružené kontextové akceptory

Iniciátor kontextu	**Příjemce kontextu**
MQCHT_CLNTCONN	FUNKCE MQCHT_SVRCONN
PŘÍJEMCE MQCHT_RECEIVER	MQCHT_SENDER
SOUBOR MQCHT_CLURCVR	MQCHT_CLUSDR

Ukončovací program zabezpečení má dva vstupní body:

- **SCSY_NTLM**

  To používá ověřovací služby NTLM, které poskytují jednosměrné ověření. NTLM umožňuje serverům ověřit identity svých klientů. Neumožňuje klientům ověřit identitu serveru nebo jeden server za účelem ověření identity jiného serveru. Ověřování NTLM bylo navrženo pro prostředí sítě, ve kterém se předpokládá, že servery jsou pravé.

- **SCY_KERBEROS**

To používá vzájemné ověřovací služby Kerberos . Protokol Kerberos nepředpokládá, že servery v síťovém prostředí jsou skutečné. Strany na obou koncích síťového připojení mohou ověřit identitu druhé strany. To znamená, že servery mohou ověřit identitu klientů a jiných serverů a klienti mohou ověřit identitu serveru.

Co dělá uživatelská procedura zabezpečení

Toto téma popisuje, co programy ukončení kanálu SSPI dělají.

Dodané uživatelské programy kanálu poskytují buď jednosměrné, nebo dvoucestné (vzájemné) ověření partnerského systému, když se zavádí relace. Pro konkrétní kanál má každý uživatelský program přidružený *činitel* (podobný ID uživatele, viz "Řízení přístupu k produktu WebSphere MQ a činitelé systému Windows" na stránce 449). Spojení mezi dvěma ukončovacími programy je asociací mezi dvěma činiteli.

Po navázání základní relace se ustavuje zabezpečené spojení mezi dvěma programy zabezpečení (jedna pro odesílající agenta MCA a jedna pro přijímajícího agenta MCA). Posloupnost operací je následující:

1. Každý program je přidružen ke konkrétnímu činiteli, například jako výsledek operace explicitního přihlášení.
2. Inicializátor kontextu vyžaduje zabezpečené připojení k partnerovi z balíku zabezpečení (pro Kerberos, jmenovaného partnera) a přijímá token (s názvem token1). Token se odešle s použitím základní relace, která je již zavedena, do partnerského programu.
3. Partnerský program (příjemce kontextu) předává token1 do balíku zabezpečení, který ověřuje, že iniciátor kontextu je autentický. Pro NTLM je nyní navázáno spojení.
4. Pro uživatelskou proceduru zabezpečení dodaný Kerberos(tj. pro vzájemné ověření), vygeneruje balík zabezpečení také druhý token (s názvem token2), který se služba kontextového příjemce vrátí do kontextového iniciátoru pomocí základní relace.
5. Inicializátor kontextu používá token2 k ověření, že je služba Client Acceptor autentická.
6. V této fázi, pokud jsou obě aplikace splněny autentičnosti tokenu partnera, je ustanoveno zabezpečené (ověřené) spojení.

Řízení přístupu k produktu WebSphere MQ a činitelé systému Windows

Řízení přístupu, které produkt WebSphere MQ poskytuje, je založeno na uživateli a skupině. Ověření, které systém Windows poskytuje, je založeno na činitelích, jako je například uživatel a servicePrincipalNázev (SPN). V případě názvu servicePrincipalmůže být mnoho z těchto přidružených k jednomu uživateli.

Uživatelská procedura zabezpečení SSPI používá příslušné řídicí služby systému Windows k ověření. Je-li ověření systému Windows úspěšné, uživatelská procedura předá ID uživatele, které je přidružené k činiteli systému Windows , pro řízení přístupu WebSphere MQ .

Řídicí služby Windows , které jsou důležité pro ověření, se liší v závislosti na typu použitého ověření.

- Pro ověření NTLM je činitel systému Windows pro kontext Context Initiator ID uživatele přidruženého k procesu, který je spuštěn. Protože je toto ověření jednosměrné, činitel přidružený k objektu Context acceptor je irelevantní.
- Pro ověření Kerberos na kanálu CLNTCONN je činitel systému Windows ID uživatele přidruženého k procesu, který je spuštěný. Jinak je činitel systému Windows názvem servicePrincipalnázvem, který je vytvořen přidáním následující předpony do názvu QueueManager.

```
ibmMQSeries/
```

Použití služeb protokolu LDAP (Lightweight Directory Access Protocol) s produktem WebSphere MQ for Windows

Toto téma vysvětluje, co je adresářová služba a část, kterou hraje protokol DAP (directory access protocol). Vysvětluje také, jak aplikace WebSphere MQ mohou používat adresář Lightweight Directory Access Protocol (LDAP) pomocí ukázkového programu jako průvodce.

Poznámka: Ukázkový program je určen pro někoho, kdo je již obeznámen s LDAP.

Následující témata poskytují více informací o adresářových službách, LDAP a použití LDAP s produktem WebSphere MQ.

- "Adresářová služba" na stránce 450
- "Lightweight Directory Access Protocol (protokol LDAP)" na stránce 450
- "Použití protokolu LDAP s produktem WebSphere MQ" na stránce 451

Adresářová služba

Adresář je úložiště informací o objektech, které je uspořádáno takovým způsobem, že je snadné najít informace o specifickém objektu.

Běžným příkladem je telefonní adresář, kde jsou informace (adresa a telefonní číslo) uloženy o lidech a firmách. Dalším příkladem je seznam adres pro e-mailový systém, kde jsou pro osoby uloženy e-mailové adresy a volitelně i další informace, jako jsou například telefonní čísla.

V počítačových systémech mohou adresáře ukládat informace o počítačových prostředcích, jako jsou tiskárny nebo sdílené disky. Například byste mohli použít adresář k vyhledání toho, kde je umístěna nejbližší barevná tiskárna. V aplikaci WebSphere MQ lze použít adresář k poskytnutí přidružení mezi aplikační službou (například zpracováním pohledávek) a frontou, která má být použita pro zprávy vyžadující danou službu (pravděpodobně identifikovanou pomocí názvu fronty a názvu správce front hostitele).

Adresáře se implementují jako systémy klient-server, kde adresářový server uchovává všechny informace a odpovídá na požadavky klientů. Klienti mohou být programy uživatelského rozhraní, které poskytují informace přímo uživateli, nebo aplikační programy, které potřebují vyhledat prostředky k dokončení jejich práce. Adresářová služba se skládá z adresářového serveru, administrativních programů a knihoven klienta a programů, které jsou potřeba ke konfiguraci, aktualizaci a čtení adresáře.

Lightweight Directory Access Protocol (protokol LDAP)

Existuje mnoho adresářových služeb, jako jsou Novell Directory Services, DCE Cell Directory Service, Banyan StreetTalk, Windows Directory Services, X.500a služby adresáře přidružené k e-mailovým produktům. X.500 byl navržen jako standard pro globální adresářové služby podle International Standards Organization (ISO). K jeho sdělení je zapotřebí protokolový zásobník OSI a do značné míry proto, že jeho užívání bylo omezeno na velké organizace a akademické instituce. Adresářový server X.500 komunikuje se svými klienty pomocí protokolu DAP (Directory Access Protocol).

Protokol LDAP (Lightweight Directory Access Protocol) byl vytvořen jako zjednodušená verze DAP. Je snadnější implementovat, vynechává některé z nejpoužívanších vlastností DAP a spouští se přes TCP/IP. V důsledku těchto změn je pro většinu účelů rychle adoptováno jako adresář pro přístup k adresáři a nahrazuje se tím množství použitých proprietárních protokolů dříve použitých. Klienti LDAP mohou stále přistupovat k serveru X.500 prostřednictvím komunikační brány (X.500 stále vyžaduje zásobník protokolu OSI) nebo stále více implementací X.500 obvykle zahrnuje nativní podporu pro LDAP a také přístup DAP.

Adresáře LDAP mohou být distribuovány a mohou pomocí replikace umožnit efektivní přístup k jejich obsahu.

Podrobnější popis protokolu LDAP najdete v tématu *Základní informace o protokolu LDAP*, publikace IBM Redbooks .

Použití protokolu LDAP s produktem WebSphere MQ

V konfiguracích produktu WebSphere MQ jsou informace, které definují frontu zpráv a přenosové fronty, uloženy lokálně. To znamená, že v síti produktu WebSphere MQ jsou distribuovány různé definice, přičemž není k dispozici žádný centrální adresář těchto informací, které jsou k dispozici pro procházení. Vzdálený systém zpráv mezi aplikacemi produktu WebSphere MQ je běžně dosažen pomocí lokálních definic vzdálených front. Aplikace nejprve vyvolá volání MQOPEN s použitím názvu zadaného v lokální definici vzdálené fronty. Chcete-li zprávu vložit do vzdálené fronty, pak aplikace vydá příkaz MQPUT, který určuje manipulátor vrácený z volání MQOPEN. Definice vzdálené fronty poskytuje název cílové fronty, správce cílové fronty a volitelně také přenosovou frontu. V této technice musí aplikace za běhu programu vědět, jak název je zadán v definici lokální fronty.

Varianta na předchozí vyhýbá se použití lokálních definic vzdálených front. Aplikace může určit úplný název cílové fronty, který obsahuje název vzdáleného správce front jako součást operace MQOPEN. Aplikace proto musí tyto dva názvy znát za běhu. Lokální správce front musí být správně konfigurován s definicí lokální fronty a s vhodně pojmenovanou (nebo výchozí) přenosovou frontou a s přidruženým kanálem, který doručuje cíli.

V případě, že jsou jak zdrojový, tak cílový správce front definován jako členy stejného klastru, lze ignorovat přenosové fronty a aspekty kanálu předchozích dvou scénářů. Je-li cílová přenosová fronta frontou klastru, lokální definice vzdálené fronty není také povinná. Nicméně podobně jako v předchozích popsaných případech musí aplikace i nadále znát název cílové fronty.

Adresářovou službu lze použít k odebrání této závislosti aplikací na názvech front (nebo v kombinaci názvů správce front a front). Mapování mezi kritérii aplikace a názvy objektů produktu WebSphere MQ lze uložit do adresáře a aktualizovat dynamicky a nezávisle na aplikacích. Při běhu aplikace WebSphere MQ , která chce odeslat zprávu, se nejprve dotáže do adresáře pomocí kritérií založených na aplikaci, například kde: service_name = "accounts pohledávek", načte relevantní názvy objektů produktu WebSphere MQ a potom použije tyto vrácené hodnoty v rámci volání MQOPEN.

Další příklad použití adresáře je určen pro společnost s mnoha malými depoty nebo kancelářemi WebSphere MQ MQI lze použít k odeslání zpráv na servery WebSphere MQ umístěné ve větších kancelářích. Klienti musí znát název hostitelského počítače, kanálu MQI a názvu fronty pro každý server, kam odesílají zprávy. Někdy může být nezbytné přesunout server WebSphere MQ do jiného počítače; každý klient, který komunikuje se serverem, bude potřebovat informace o této změně. Adresářová služba LDAP by mohla být použita k ukládání názvů hostitelských počítačů (a názvy kanálů a front) a klientské programy by mohly načíst informace z adresáře, kdykoli chtějí odeslat zprávu na server. V tomto případě je třeba aktualizovat pouze adresář, pokud došlo ke změně názvu hostitele (nebo kanálu nebo názvu fronty).

V adresáři může být uloženo více míst určení pro zprávu aplikace, přičemž jedna z vybraných cílů bude záviset na dostupnosti nebo úvahách o sdílení zátěže.

Produkt WebSphere MQ může také použít adresář LDAP k ukládání ověřovacích informací pro použití se zabezpečením SSL (Secure Sockets Layer). Třídy WebSphere MQ pro prostředí Java mohou rovněž ukládat informace do adresáře LDAP.

Ukázkový program LDAP

Ukázkový program je určen pro uživatele, který je obeznámen s protokolem LDAP a pravděpodobně jej již používá. Cílem je ukázat, jak produkt WebSphere MQ může používat adresář LDAP.

Sestavení ukázkového programu

Tento program byl sestaven a testován pouze v systému Windows pomocí protokolu TCP/IP. Stejně jako obecné úvahy uvedené v části “Příprava programů jazyka C v systému Windows” na stránce 443si všimněte následujících bodů:

- Tento program je navržen tak, aby se spouštěl jako klientský program, takže by měl být propojen s rozhraním MQIC.LIB .

- Stejně jako soubory záhlaví a knihovny produktu WebSphere MQ musí být tento program sestaven s použitím souborů a knihoven záhlaví klienta LDAP.

  Používáte-li například klienta IBM eNetwork , propojte program s LIBLDAPSTATICE.LIB a LIBLBERSTATICSSL.LIB .

Konfigurace adresáře

Než bude možné spustit ukázkový program, musí být adresářový server LDAP konfigurován s ukázkovými daty.

Soubor MQuser.ldif, v adresáři `tools\c\samples` , obsahuje některá ukázková data ve formátu LDIF (LDAP Data Interchange Format). Tento soubor můžete upravit tak, aby vyhovoval vašim potřebám. Obsahuje údaje o fiktivní společnosti s názvem MQuser, která má oddělení pro dopravu obsahující tři kanceláře. Každý z těchto kanceláří má počítač, na kterém je spuštěn server WebSphere MQ .

Jako minimum musíte upravit tři řádky, které obsahují názvy hostitelů na počítačích se servery WebSphere MQ : řádky 18, 27 a 36:

```
host: LondonHost
  ...
host: SydneyHost
  ...
host: WashingtonHost
```

Musíte změnit `LondonHost`, `SydneyHosta` `WashingtonHost` na názvy tří svých počítačů, které spouštějí servery WebSphere MQ . Chcete-li (ukázku používá názvy výchozích hodnot systému), můžete také změnit názvy kanálů a front. Možná budete chtít také zvýšit nebo snížit počet kanceláří v ukázkových datech.

Konfigurace adresářového serveru IBM Tivoli

Informace o instalaci adresáře najdete v příručce IBM Tivoli Directory Server (ITDS) Administrator's Guide. V tématu `Installing and Configuring Server`pracujte v sekcích `Installing Server` a `Basic Server Configuration`. V případě potřeby si přečtěte téma `Administrator Interface` a seznamte se s tím, jak funguje rozhraní.

V tématu `Configuring - How Do I`postupujte podle pokynů pro spuštění správce, potom pracujte v sekci `Configure Database` a vytvořte výchozí databázi. Přeskočte sekci `Configure replica` a použijete sekci `Work with Suffixes`, přidejte příponu `o=MQuser`.

Před přidáním jakýchkoli záznamů do databáze musíte rozšířit schéma adresáře přidáním některých definic atributu a definice třídy objektu. To je popsáno v příručce IBM Tivoli Directory Server Administrator's Guide v kapitole `Reference Information` pod sekcí `Directory Schema`. K dispozici jsou dva ukázkové soubory, které vám s tím pomohou. Soubor `mq.at.conf` obsahuje definice atributů, které musíte přidat do souboru `?etc?slapd.at.conf`. Toto proveďte zahrnutím ukázkového souboru úpravou `slapd.at.conf` a přidáním řádku:

```
include <pathname>/mq.at.conf
```

Případně můžete upravit soubor `slapd.at.conf` a přidat obsah ukázkového souboru přímo do něj, tj. přidat řádky:

```
# MQ attribute definitions
attribute mqChannel           ces   mqChannel        1000   normal
attribute mqQueueManager      ces   mqQueueManager   1000   normal
attribute mqQueue             ces   mqQueue          1000   normal
attribute mqPort              cis   mqPort           64     normal
```

Podobně jako u definice třídy objektu můžete buď zahrnout ukázkový soubor úpravou souboru `etc?slapd.oc.conf` a přidáním řádku:

```
include <pathname>/mq.oc.conf
```

nebo můžete přidat obsah ukázkového souboru přímo do produktu `slapd.oc.conf,` , který je, přidejte řádky:

```
# MQ object classdefinition
objectclass mqApplication
      requires
              objectClass,
              cn,
              host,
              mqChannel,
              mqQueue
      allows
              mqQueueManager,
              mqPort,
              description,
              l,
              ou,
              seeAlso
```

Nyní můžete spustit adresářový server (Administration, Server, Startup) a přidat do něj vzorové záznamy. Chcete-li přidat ukázkové položky, přejděte na stránku Administrace, Přidat položky administrátora, zadejte úplnou cestu k ukázkovému souboru `MQuser.ldif` a klepněte na tlačítko Odeslat.

Adresářový server je nyní spuštěn a načten s daty vhodnými pro spuštění ukázkového programu.

Konfigurace adresářového serveru Netscape

Na stránce administrace serveru Netscape klepněte na volbu **Vytvořit nový server Netscape Directory Server**.

Nyní byste měli být prezentovány s formulářem obsahujícím informace o konfiguraci. Změňte příponu adresáře na **o = MQuser** a přidejte heslo pro uživatele bez omezení. Můžete také změnit jakékoli jiné informace, které budou vyhovovat vaší instalaci. Klepněte na tlačítko **OK**a adresář by měl být úspěšně vytvořen. Klepněte na tlačítko **Návrat na administraci serveru** a spusťte adresářový server. Klepněte na název adresáře, chcete-li spustit server Directory Server Administration pro nový adresář.

Před přidáním jakýchkoli záznamů do databáze rozšiřte schéma adresáře přidáním některých definic atributu a definice třídy objektu. Klepněte na kartu **Schéma** na stránce Adresářový server. Nyní se vám nabídne formulář, který vám umožňuje přidávat nové atributy. Přidejte následující atributy (ponechte prázdné pole atributu pro všechny z nich):

```
Attribute Name              Syntax
--------------              ------
mqChannel                   Case Exact String
mqQueueManager              Case Exact String
mqQueue                     Case Exact String
mqPort                      Integer
```

Přidejte novou položku objectClass klepnutím na volbu **Vytvořit třídu ObjectClass** na postranním panelu. Zadejte **mqApplication** jako ObjectClass Název, vyberte **applicationProcess** jako nadřízený objekt ObjectClass a ponecháte pole **ObjectClass OID** prázdné. Nyní přidejte některé atributy do třídy objectClass. Vyberte volby **host**, **mqChannel**a **mqQueue** jako povinné atributy a vyberte volbu **mqQueueManager** a **mqPort** jako Povolené atributy. Stisknutím tlačítka **Vytvořit novou třídu objektů ObjectClass** vytvořte objekt objectClass.

Chcete-li přidat ukázková data, klepněte na kartu **Správa databáze** a z postranního panelu vyberte volbu **Přidat položky** . Zadejte název cesty ukázkového datového souboru `<pathname>\MQuser.ldif`, zadejte heslo a klepněte na tlačítko **OK**.

Ukázkový program je spuštěn jako neautorizovaný uživatel a ve výchozím nastavení adresář Netscape neumožňuje neautorizovaným uživatelům prohledávat adresář. Tuto změnu můžete změnit klepnutím na

kartu **Řízení přístupu** . Zadejte heslo pro uživatele bez omezení a klepněte na tlačítko **OK** , abyste se načetli do položek řízení přístupu pro daný adresář. Ty by měly být v současné době prázdné. Chcete-li vytvořit novou položku řízení přístupu, stiskněte tlačítko **Nová položka ACI** . V zobrazeném poli klepněte na tlačítko **Odepřít** (podtržené) a ve výsledném dialogovém okně změňte hodnotu na **Povolit**. Přidejte název, například **MQuser-access**, a klepněte na volbu **vybrat příponu** a vyberte volbu **o = MQuser**. Zadejte hodnotu **o = MQuser** jako cíl, zadejte heslo pro nevyhrazený uživatel a klepněte na tlačítko **Odeslat**.

Adresářový server je nyní spuštěn a načten s daty vhodnými pro spuštění ukázkového programu.

Spuštění ukázkového programu

Nyní byste měli mít adresářový server LDAP spuštěný a naplněný ukázkovými daty. Tato data uvádí tři hostitelské počítače, z nichž všechny by měly běžet na serverech WebSphere MQ . Ujistěte se, že je výchozí správce front spuštěn na každém počítači (pokud jste nezměnili vzorová data pro uvedení jiného správce front).

Také spusťte program modulu listener produktu WebSphere MQ na každém počítači; ukázka používá protokol TCP/IP s výchozím číslem portu produktu WebSphere MQ , takže můžete modul listener spustit pomocí příkazu:

```
runmqlsr -t tcp
```

Chcete-li otestovat ukázku, můžete také chtít spustit program ke čtení zpráv přicházejících na každý server WebSphere MQ , například můžete použít ukázkový program amqstrg:

```
amqstrg SYSTEM.DEFAULT.LOCAL.QUEUE
```

Ukázkový program používá tři proměnné prostředí, jeden povinný a dva volitelné. Požadovaná proměnná je LDAP_BASEDN, která uvádí základní rozlišující název pro vyhledávání adresáře. Chcete-li pracovat s ukázkovými daty, nastavte tuto hodnotu na `ou=Transport, o=MQuser`, například na příkazový řádek na typu systémů Windows :

```
set LDAP_BASEDN=ou=Transport, o=MQuser
```

Nepovinné proměnné jsou LDAP_HOST a LDAP_VERSION. Proměnná LDAP_HOST uvádí název hostitele, na kterém je spuštěn server LDAP; standardně se použije na lokálního hostitele, pokud není zadán. Proměnná LDAP_VERSION uvádí verzi protokolu LDAP, která má být použita, a může být buď 2, nebo 3. Většina serverů LDAP nyní podporuje verzi 3 protokolu; všechny podporují starší verzi 2. Tato ukázka funguje stejně dobře s verzí protokolu, a pokud není uvedena, použije se výchozí hodnota verze 2.

Nyní můžete ukázku spustit zadáním názvu programu následovaného názvem aplikace produktu WebSphere MQ , do níž chcete odesílat zprávy, v případě ukázkových dat jsou názvy aplikací Londýn, Sydney a Washington. Chcete-li například odeslat zprávy do aplikace Londýn, postupujte takto:

```
amqsldpc London
```

Pokud se programu nepodaří připojit k serveru WebSphere MQ , zobrazí se příslušná chybová zpráva. Pokud se vám podaří úspěšně začít psát zprávy, každý řádek, který zadáte (končí < return> nebo < enter>), se odešle jako samostatná zpráva, prázdný řádek ukončí program.

Návrh programu

Program má dvě odlišné části: první část používá proměnné prostředí a hodnotu příkazového řádku k dotazu na adresářový server LDAP; druhá část ustanoví připojení WebSphere MQ pomocí informací vrácených z adresáře a odešle zprávy.

Volání LDAP použitá v první části programu se mírně liší v závislosti na tom, zda je používána služba LDAP verze 2 nebo 3, a jsou podrobně popsány v dokumentaci dodané s knihovnami klienta LDAP. Tento oddíl obsahuje stručný popis.

První část programu kontroluje, zda byla správně volána, a čte proměnné prostředí. Poté naváže spojení s adresářovým serverem LDAP na uvedeném hostiteli:

```
if (ldapVersion == LDAP_VERSION3)
{
  if ((ld = ldap_init(ldapHost, LDAP_PORT)) == NULL)
     ...
}
else
{
  if ((ld = ldap_open(ldapHost, LDAP_PORT)) == NULL )
     ...
}
```

Když bylo navázáno spojení, program nastavuje některé volby na serveru s voláním "ldap_set_option" a ověřuje se na serveru tím, že se k němu váže:

```
if (ldapVersion == LDAP_VERSION3)
{
  if (ldap_simple_bind_s(ld, bindDN, password) != LDAP_SUCCESS)
     ...
}
else
{
  if (ldap_bind_s(ld, bindDN, password, LDAP_AUTH_SIMPLE) !=
      LDAP_SUCCESS)
     ...
}
```

V ukázkovém programu `bindDN` a `password` jsou nastaveny na hodnotu NULL, což znamená, že se program ověřuje jako anonymní uživatel, tj. nemá žádná zvláštní přístupová práva a má přístup pouze k informacím, které jsou veřejně dostupné. V praxi většina organizací omezuje přístup k informacím, které ukládají v adresářích, takže k nim mají přístup pouze autorizovaní uživatelé.

První parametr pro volání vazby `ld` je popisovač, který se používá k identifikaci této konkrétní relace LDAP během celého zbytku programu. Po ověření program prohledá adresář a hledá položky, které odpovídají názvu aplikace:

```
rc = ldap_search_s(ld,                /* LDAP Handle             */
                   baseDN,            /* base distinguished name */
                   LDAP_SCOPE_ONELEVEL, /* one-level search       */
                   filterPattern,     /* filter search pattern   */
                   attrs,             /* attributes required     */
                   FALSE,             /* NOT attributes only     */
                   &ldapResult);      /* search result           */
```

Jedná se o jednoduché synchronní volání na server, které vrací výsledky přímo. Existují další typy hledání, které jsou vhodnější pro složité dotazy nebo když se očekává velký počet výsledků. První parametr pro vyhledávání je manipulátor `ld` , který identifikuje relaci. Druhý parametr je základní rozlišující název, který uvádí, kde v adresáři má hledání začít hledat, a třetí parametr je rozsah hledání, tedy prohledávaný záznamy relativně vzhledem k výchozímu bodu. Tyto dva parametry společně definují, které záznamy v adresáři se prohledají. V dalším parametru `filterPattern` určuje, co hledáme. Parametr `attrs` obsahuje seznam atributů, které chceme získat zpět od objektu, když jsme jej našli. Následující atribut říká, zda chceme pouze atributy nebo jejich hodnoty; nastavením tohoto na FALSE znamená, že chceme hodnoty atributu. Konečný parametr se používá k vrácení výsledku.

Výsledek může obsahovat mnoho záznamů adresáře, každý se zadanými atributy a jejich hodnotami. Musíme extrahovat hodnoty, které chceme od výsledku. V tomto ukázkovém programu očekáváme, že bude nalezen pouze jeden záznam, takže se podíváme pouze na první položku ve výsledku:

```
ldapEntry = ldap_first_entry(ld, ldapResult);
```

Toto volání vrátí popisovač, který představuje první položku, a nastavíme smyčku for, aby extrahoval všechny atributy z položky:

```
for (attribute = ldap_first_attribute(ld, ldapEntry, &ber);
     attribute != NULL;
     attribute = ldap_next_attribute(ld, ldapEntry, ber ))
{
```

Pro každý z těchto atributů extrahujeme hodnoty přidružené k tomuto atributu. Opět očekáváme pouze jednu hodnotu na atribut, takže použijeme pouze první hodnotu; určujeme, který atribut máme a kam se má uložit hodnota v odpovídající proměnné programu:

```
values = ldap_get_values(ld, ldapEntry, attribute);
if (values != NULL && values[0] != NULL)
{
  if (stricmp(attribute, MQ_HOST_ATTR) == 0)
  {
    mqHost = strdup(values[0]);
    ...
```

Nakonec jsme se uklidil uvolněním paměti (ldap_value_free, ldap_memfree, ldap_msgfree) a zavřít relaci *unbinding* ze serveru:

```
ldap_unbind(ld);
```

Zjistili jsme, že jsme našli všechny hodnoty WebSphere MQ , které potřebujeme z adresáře, a pokud ano, zavoláme sendMessages() pro připojení k serveru WebSphere MQ a odešlete zprávy produktu WebSphere MQ .

Druhá část ukázkového programu je rutina sendMessages(), která obsahuje všechna volání WebSphere MQ . Toto je modelováno pomocí ukázkového programu amqsput0 , rozdíly v tom, že parametry programu byly rozšířeny, a MQCONNX se použije místo volání MQCONN.

Vývoj aplikací pro produkt IBM WebSphere MQ Telemetry

Telemetrické aplikace integrují smyslová a řídicí zařízení s dalšími zdroji informací, které jsou k dispozici na internetu a v podnicích.

Vyvíjejte aplikace pro produkt IBM WebSphere MQ Telemetry s využitím vzorů návrhu, příkladů práce, ukázkových programů, koncepcí programování a referenčních informací. Pomocí démona IBM WebSphere MQ Telemetry pro zařízení zjednodušíte připojení mnoha malých zařízení k produktu IBM WebSphere MQ.

Související pojmy

WebSphere MQ Telemetry

Koncepty a scénáře telemetrie pro monitorování a řízení

Související úlohy

Instalace produktu WebSphere MQ Telemetry

Správa produktu WebSphere MQ Telemetry

Odstraňování problémů s produktem WebSphere MQ Telemetry

Související odkazy

WebSphere MQ Telemetry -referenční informace

IBM WebSphere MQ Telemetry ukázkové programy

K dispozici jsou ukázkové skripty, které demonstrují základní použití aplikace MQ Telemetry Transport v3 Client. Pomocí skriptů publikujte zprávu a přihlaste se k odběru tématu.

Než začnete

Spusťte službu telemetrie (MQXR) a spusťte ukázkové programy.

ID uživatele musí být členem skupiny uživatelů mqm.

Nejprve spusťte skript SampleMQM , za nímž bude následovat skript MQTTV3Sample k provedení publikování a odběru. Spuštěním ukázkového skriptu CleanupMQM můžete odstranit správce front vytvořeného skriptem SampleMQM .

Protože skript SampleMQM vytváří a používá správce front s názvem QM1, v systému se správcem front QM1 nespouští žádné změny v systému. Jakékoli provedené změny mohou mít vliv na konfiguraci existujícího správce front.

Informace o této úloze

- Aplikace SampleMQM vytvoří a spustí správce front s podporou telemetrie s názvem QM1. Skript také nastaví výchozí přenosovou frontu pro QM1a vytvoří a spustí výchozí kanál naslouchající na portu 1883. Tento kanál nezajišťuje žádné ověřování klientů připojených k tomuto kanálu. Kanál má atribut Identifikátor uživatele agenta kanálu zpráv (MCAUSER), nastavený na 'guest' na systémech Windows nebo 'nobody' na systémech Linux . Klienti, kteří jsou připojeni k kanálu, jsou léčeni jako uživatel 'guest' nebo uživatel 'nobody', v závislosti na systému, na kterém je spuštěný. Skript autorizuje 'guest' na systémech Windows a 'nobody' na systémech Linux , aby bylo možné publikovat a odebírat libovolné téma v systému QM1
- Aplikace MQTTV3Sample se nachází v následujícím umístění:
  - V prostředí Windows *MQ_INSTALLATION_PATH*\mqxr\samples

    kde *INSTALAČNÍ_CESTA_PRODUKTU_MQ* je umístění, ve kterém je nainstalován produkt IBM WebSphere MQ .
  - V prostředí Linux *MQ_INSTALLATION_PATH*/mqxr/samples

  Aplikace MQTTV3Sample pracuje jako vydavatel a posílá na server jednu zprávu na téma. Také se chová jako odběratel a naslouchá zprávám ze serveru.
- Ukázkový skript CleanupMQM končí a odstraní QM1 , který byl vytvořen skriptem SampleMQM . Ukázkový skript CleanupMQM použijte v případě, že chcete znovu spustit skript SampleMQM , nebo odeberte produkt QM1.

Postup

1. Chcete-li spustit skript SampleMQM , zadejte na příkazový řádek následující příkaz:
   - V systému Windowsje příkaz ke spuštění skriptu SampleMQM následující:

     ```
     MQ_INSTALLATION_PATH\mqxr\samples\SampleMQM.bat
     ```
   - V systémech AIX a Linuxse takto označuje příkaz ke spuštění skriptu SampleMQM :

     ```
     MQ_INSTALLATION_PATH/mqxr/samples/SampleMQM.sh
     ```

   kde *INSTALAČNÍ_CESTA_PRODUKTU_MQ* je umístění, ve kterém je nainstalován produkt IBM WebSphere MQ .

   Vytvoří se správce front s názvem MQXR_SAMPLE_QM.
2. Chcete-li spustit první část skriptu MQTTV3Sample , zadejte následující příkaz;
   - V systému Windowszadejte na jednom příkazovém řádku tento příkaz:

     ```
     MQ_INSTALLATION_PATH\mqxr\samples\RunMQTTV3Sample.bat -a subscribe
     ```
   - V systému AIX a Linuxzadejte do jednoho okna shellu následující příkaz:

     ```
     MQ_INSTALLATION_PATH/mqxr/samples/RunMQTTV3Sample.sh -a subscribe
     ```
3. Chcete-li spustit druhou část skriptu MQTTV3Sample , zadejte následující příkaz:

- V systému Windowszadejte na jiný příkazový řádek následující příkaz:

```
MQ_INSTALLATION_PATH\mqxr\samples\RunMQTTV3Sample.bat -m "Hello from an MQTT v3
application"
```

- V systémech AIX a Linuxzadejte do jiného okna shellu následující příkaz:

```
MQ_INSTALLATION_PATH/mqxr/samples/RunMQTTV3Sample.sh -m "Hello from an MQTT v3 application"
```

4. Chcete-li odebrat správce front vytvořeného pomocí skriptu SampleMQM , můžete spustit skript CleanupMQM pomocí následujícího příkazu:
   - V systému Windowszadejte následující příkaz:

```
MQ_INSTALLATION_PATH\mqxr\samples\CleanupMQM.bat
```

   - V systémech AIX a Linux v jiném okně shellu zadejte následující příkaz:

```
MQ_INSTALLATION_PATH/mqxr/samples/CleanupMQM.sh
```

Výsledky

Zpráva `Hello from an MQTT v3 application` , kterou jste zadali do druhého okna, bude publikována touto aplikací a bude přijata aplikací v prvním okně. Aplikace v prvním okně se zobrazí na obrazovce.

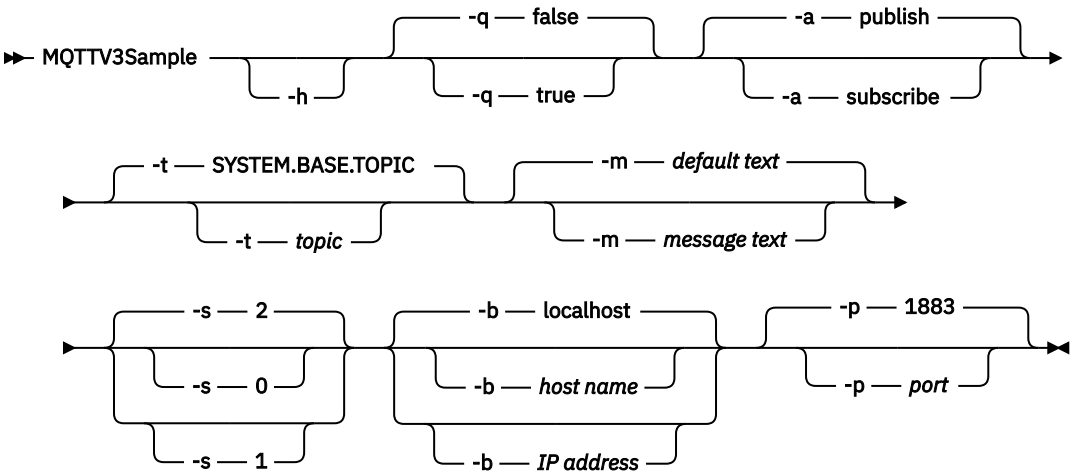
Program MQTTV3Sample

Referenční informace o ukázkové syntaxi a parametrech pro program MQTTV3Sample .

Účel

Program MQTTV3Sample lze použít k publikování zprávy a k přihlášení k odběru tématu.

MQTTV3Sample syntax



Parametry

-h
Vytisknout tento text nápovědy a ukončit

-q
Nastavte tichý režim místo použití výchozího režimu false.

-a
Nastavte publikování nebo odběr, místo toho, že byste předpokládali výchozí akci publikování.

-t
Publikování nebo odběr tématu, místo publikování nebo odběru výchozího tématu

-m
Publikujte text zprávy místo odeslání výchozího textu publikace, "Ahoj z aplikace MQTT v3 ".

-s
Nastavte QoS místo použití předvolené QoS, 2.

-b
Připojte se k tomuto názvu hostitele nebo k adrese IP namísto připojení k výchozímu názvu hostitele localhost.

-p
Použijte tento port místo použití výchozího nastavení, 1883.

Spusťte program MQTTV3Sample

Chcete-li se přihlásit k odběru tématu v systému Windows, použijte tento příkaz:

```
runMQTTV3Sample -a subscribe
```

Chcete-li publikovat zprávu v systému Windows, použijte příkaz:

```
runMQTTV3Sample
```

Další informace o spuštění poskytnutých ukázkových skriptů viz “IBM WebSphere MQ Telemetry ukázkové programy” na stránce 456.

Vytvoření první aplikace MQ Telemetry Transport Publisher pomocí jazyka Java

Kroky k vytvoření aplikace klienta MQTT jsou popsány ve výukovém programu. Každý řádek kódu je vysvětlen. Na konci úlohy jste vytvořili vydavatele MQTT. Příručky můžete procházet pomocí průzkumníka WebSphere MQ Explorer.

Než začnete

Nainstalujte funkci WebSphere MQ Telemetry na server, který má nainstalován produkt IBM WebSphere MQ Version 7.1 nebo novější.

Klientská aplikace používá balík produktu `com.ibm.mq.micro.client.mqttv3` v sadě nástrojů Software Development Toolkit (SDK) produktu IBM WebSphere MQ Telemetry . Sada SDK je součástí instalace produktu IBM WebSphere MQ Telemetry . Klient se připojuje k funkci IBM WebSphere MQ Telemetry pro výměnu zpráv s IBM WebSphere MQ.

Chcete-li spravovat produkt IBM WebSphere MQ Telemetry, musíte také nainstalovat aktualizace telemetrie pro produkt IBM WebSphere MQ Explorer Version 7.1 . Aktualizace jsou součástí instalace produktu IBM WebSphere MQ Telemetry .

Klient MQTT spuštěný v prostředí Java SE vyžaduje verzi 6.0 jazyka Java SE nebo novější. Produkt IBM Java SE v6.0 je součástí instalace produktu IBM WebSphere MQ Version 7.1 . Nachází se v *WebSphere MQ installation directory*`\java\jre`

Informace o této úloze

Příkladem je publikační aplikace, `PubSync`. Produkt `PubSync` publikuje `Hello World` na téma `MQTT Examples`a čeká na potvrzení, že publikování bylo doručeno správci front.

Nastavením trvalého odběru na produkt `MQTT Examples` můžete zkontrolovat, zda tato aplikace pracuje.

Procedura používá platformu Eclipse k vývoji, sestavení a spuštění klienta. Eclipse si můžete stáhnout z webu projektu Eclipse na adrese www.eclipse.org.

Chcete-li vytvořit aplikaci, můžete vytvořit soubory Java a kompilovat je a spustit je pomocí příkazového řádku.

V novém adresáři vytvořte cestu k adresáři `.\com\ibm\mq\id`. Vytvořte dva soubory Java, `Example.java` a `PubSync.java`. Zkopírujte kód z produktu "Příklad kódu" na stránce 463 do souborů Java.

Kompilace kódu Java pomocí příkazu,

```
javac -cp jar_dir\com.mq.micro.client.mqttv3.jar
       com.ibm.mq.id.PubSync.java com.ibm.mq.id.Example.java
```

Spusťte příkaz `PubSync` pomocí příkazu,

```
java -cp jar_dir\com.mq.micro.client.mqttv3.jar
      com.ibm.mq.id.PubSync
```

Postup

1. Vytvořte projekt Java v prostředí Eclipse.
   a) **Soubor > Nový > Projekt Java** a zadejte název projektu. Klepněte na tlačítko **Další**.

      Zkontrolujte, zda je prostředí JRE správné nebo pozdější verze. Java SE musí být na 6.0 nebo pozdější.

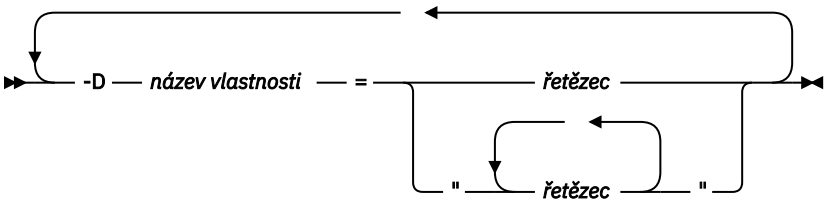
   b) Na stránce Nastavení Java klepněte na volbu **Knihovny > Přidat externí soubory JAR ...**
   c) Přejděte do adresáře, do kterého jste nainstalovali složku WebSphere MQ Telemetry SDK. Vyhledejte složku `SDK\clients\java` a vyberte všechny soubory `.jar` > **> Otevřít > Dokončit**.
2. Nainstalujte dokumentaci Javadoc klienta protokolu MQTT.

   Je-li instalována dokumentace Javadoc klienta protokolu MQTT, poskytuje editor jazyka Java pomoc s třídami MQTT v3 .

   a) Ve svém projektu Java otevřete **Průzkumník balíků > Odkazované knihovny**. Klepněte pravým tlačítkem myši na položku `com.ibm.micro.client.mqttv3.jar` **> Vlastnosti**.
   b) V navigátoru Vlastnosti klepněte na volbu **Umístění dokumentace Javadoc**.
   c) Na stránce Umístění dokumentace Javadoc klepněte na **Adresa URL dokumentace Javadoc > Procházet ...** a vyhledejte složku *`WMQ Installation directory`*`\mqxr\SDK\clients\java\doc\javadoc` **> OK**.
   d) Klepněte na tlačítko **Ověřit ... > OK** .

      Zobrazí se výzva k otevření prohlížeče a zobrazení dokumentace.
3. Vytvořte třídu `PubSync`pomocí průvodce třídou Java.
   a) Klepněte pravým tlačítkem myši na projekt Java, který jste vytvořili > **Nový** > **Třída**.
   b) Zadejte název balíku, `com.ibm.mq.id`
   c) Zadejte název třídy, `PubSync`
   d) Zkontrolujte stub metody metody **public static void main (String [] args)**
4. Vytvořte soubor, `Example.java` v balíku `com.ibm.mq.id`. Zkopírujte kód do souboru Obrázek 89 na stránce 465 do souboru.

   Všechny parametry použité v příkladech jsou nastaveny jako vlastnosti. Hodnoty lze přepsat změnou výchozích hodnot v produktu `Example.java`nebo zadáním vlastností jako voleb na příkazový řádek jazyka Java pomocí parametru `-D` :



Identifikátor klienta použitý v tomto příkladu a příklady “Vytvoření asynchronního vydavatele pro přenos MQ Telemetry pomocí jazyka Java” na stránce 465 je jméno uživatele s příponou náhodným řetězcem.

5. Postupujte podle kroků k vytvoření kódu, nebo zkopírujte kód z produktu Obrázek 88 na stránce 464.

   Kroky, které následují, vysvětlují kód v produktu `Pubsync.java`.

6. Vytvořte blok `try-catch` .

```
try { ...
 } catch (Exception e) {
  e.printStackTrace();
}
```

   Klient MQTT generuje `MqttException`, `MqttPersistenceException` nebo `MqttSecurityException`. `MqttPersistenceException` a `MqttSecurityException` jsou podtřídy `MqttException`.

   Chcete-li zjistit příčinu výjimky, použijte metodu produktu `MqttException.getReasonCode` . Je-li `MqttPersistenceException` nebo `MqttSecurityException` hozena, použijte metodu `getCause` k vrácení podkladové události typu throwable.

7. Vytvoření nové instance `MqttClient` .

```
MqttClient client = new MqttClient(Example.TCPAddress, Example.clientId);
```

   Zadejte klienta s adresou serveru, která se používá později k připojení k produktu WebSphere MQ. Nastavte identifikátor klienta tak, aby pojmenujete klienta.

   - Volitelně můžete poskytnout implementaci rozhraní produktu `MqttClientPersistence` , která nahradí výchozí implementaci. Výchozí implementace produktu `MqttPersistence` ukládá QoS 1 a 2 zprávy čekající na doručení jako soubory; viz “Perzistence zpráv v klientech produktu MQTT” na stránce 516.
   - The default IBM WebSphere MQ TCP/IP port for MQTT is 1883. Pro zabezpečení SSL je to 8883. V tomto příkladu je výchozí adresa nastavena na `tcp://localhost:1883`.
   - Obvykle je důležité, abyste mohli identifikovat specifického fyzického klienta pomocí identifikátoru klienta. Identifikátor klienta musí být jedinečný v rámci všech klientů, kteří se připojují k serveru; viz “Identifikátor klienta” na stránce 512. Použití stejného identifikátoru klienta jako předchozí instance označuje, že aktuální instance je instancí stejného klienta. Pokud duplikujete identifikátor klienta ve dvou spuštěných klientech, dojde k výjimce v obou klientech a jeden z klientů se ukončí.
   - Délka identifikátoru klienta je omezena na 23 bajtů. Dojde-li k překročení délky, dojde k výjimce. Identifikátor klienta musí obsahovat pouze znaky povolené v názvu správce front, například bez pomlček nebo mezer.
   - Dokud nezavoláte metodu `MqttClient.connect` , žádné zpracování zpráv se neprovádí.

   Pomocí objektu klienta můžete publikovat a odebírat témata a obnovovat informace o publikatech, které dosud nebyly doručeny.

8. Vytvořte téma pro publikování.

```
MqttTopic topic = client.getTopic(Example.topicString);
```

   Délka řetězce tématu je omezena na 64 kB, což překračuje maximální délku řetězce tématu IBM WebSphere MQ . V opačném případě bude řetězec tématu postupovat podle stejných pravidel jako

řetězce témat produktu WebSphere MQ . Další informace naleznete v tématu Řetězce témat. Příklad nastavuje řetězec tématu `MQTT Examples`.

9. Vytvořte publikační zprávu.

```
MqttMessage message = new MqttMessage(Example.publication.getBytes());
```

Řetězec `"Hello World"` se převede na bajtové pole a použije se k vytvoření `MqttMessage`.

- Informační obsah zprávy MQTT je vždy bajtové pole. Metoda `getBytes` převádí řetězcový objekt na UTF-8. Produkt `MqttMessage` má pohodlnější metodu `toString` , která vrací informační obsah zprávy jako řetězec. Je ekvivalentní s `new string(message.getPayload)`
- Do správce front se odešle zpráva o publikování se záhlavím RFH2 a data zprávy jsou odeslána jako zpráva `jms-bytes` .
- Objekt zprávy má kvalitu služby a uchované atributy. Kvalita služby (QoS) určuje, jak spolehlivě je zpráva přenášena mezi klientem MQTT a správcem front; viz “Kvality služby poskytované klientem MQTT” na stránce 520. Zachovaný atribut řídí, zda je publikování uloženo správcem front pro budoucí odběratele. Pokud se publikování nezachová, odešle se pouze aktuálním odběratelům; viz “Zachovaná publikování a klienti MQTT” na stránce 522. Standardní nastavení `MqttMessage` jsou "Zprávy se doručují alespoň jednou a neuchovávají se."

10. Připojte se k serveru.

```
client.connect();
```

Příklad se připojí k serveru s použitím výchozích voleb připojení. Jakmile se připojíte, můžete začít publikovat. Výchozí volby připojení jsou:

- Každých 15 sekund se odešle malá zpráva "keep-alive", aby se zabránilo zavření připojení TCP/IP.
- Relace se spustí bez kontroly dokončení předchozích publikování.
- Interval mezi pokusy o odeslání zprávy je opět 15 sekund.
- Pro připojení se nevytvoří žádná poslední zpráva a zpráva o potvrzení.
- K vytvoření připojení se používá standardní hodnota SocketFactory .

Změňte volby připojení tak, že vytvoříte objekt `ConnectionOptions` a předáte jej jako další parametr pro `client.connect`.

11. Publish.

```
MqttDeliveryToken token = topic.publish(message);
```

Příklad odešle do správce front publikování "Ahoj světe" na téma "Příklady MQTT".

- Když se metoda `publish` vrátí, zpráva je bezpečně přenesena do klienta MQTT, ale ještě nebyla přenesena na server. Pokud má zpráva QoS 1 nebo 2, pak je zpráva uložena lokálně, v případě selhání klienta před dokončením doručení.
- `publish` vrací token doručení, který se používá ke kontrole, zda bylo přijato potvrzení od serveru.

12. Čekejte na potvrzení ze serveru.

```
token.waitForCompletion(Example.timeout);
```

Příklad PubSync čeká na potvrzení ze serveru, což potvrzuje, že zpráva byla doručena.

- Bez vypršení časového limitu bude klient čekat nekonečně dlouhou dobu. Úloha “Vytvoření asynchronního vydavatele pro přenos MQ Telemetry pomocí jazyka Java” na stránce 465 zobrazuje způsob příjmu potvrzení bez čekání na použití objektu zpětného volání.

13. Odpojte klienta od serveru.

```
client.disconnect();
```

Klient se odpojí od serveru a čeká na všechny metody `MqttCallback` , které jsou spuštěny, aby bylo možné dokončit. Poté čeká až 30 sekund na dokončení zbývající práce. Jako další parametr můžete uvést časový limit uvedení do klidového stavu.

14. Uložit změny do `PubSync.java` a `Example.Java`

    Platforma Eclipse automaticky kompiluje prostředí Java. Nyní jste připraveni se podívat na výsledky spuštěním programu.

Výsledky

Chcete-li si prohlédnout publikace pomocí produktu WebSphere MQ, vytvořte téma, frontu a trvalý odběr, všechny nazývané `"MQTTExampleTopic"` používající skript v produktu Obrázek 87 na stránce 463. Spusťte klienta pro publikování na téma `MQTT Examples` a poté spusťte ukázkový program **amqsbcg** a procházejte publikace ve frontě produktu `MQTTExamples` .

1. Spusťte správce front a spusťte službu telemetrie (MQXR) spuštěnou. Ujistěte se, že adresa TCP/IP a port konfigurovaný pro kanál telemetrie se shodují s hodnotami, které používáte v aplikaci MQTT.
2. Nakonfigurujte trvalý odběr vytvořením příkazového skriptu `mqttexamples.txt` a jeho spuštěním s použitím produktu **runmqsc:** .

```
DEFINE TOPIC('MQTTExampleTopic') TOPICSTR('MQTT Example') REPLACE
DEFINE QLOCAL('MQTTExampleQueue') REPLACE
DEFINE SUB('MQTTExampleSub') DEST('MQTTExampleQueue') TOPICOBJ('MQTTExampleTopic') REPLACE
```

Obrázek 87. `mqttExampleTopic.txt`

Chcete-li spustit skript v systému Windows, zadejte následující příkaz:

```
runmqsc queue manager name < mqttExampleTopic.txt
```

3. Spusťte klienta jako aplikaci Java z platformy Eclipse, nebo spuštěním jazyka Java v příkazovém okně:

```
java -cp jar_dir\com.mq.micro.client.mqttv3.jar
      com.ibm.mq.id.classname.class
```

Poznámka: Příkazové okno musí být otevřeno v adresáři, který obsahuje cestu, `com\ibm\mq\id`.

4. Procházejte výsledky pomocí průzkumníka WebSphere MQ Explorer nebo spusťte příkaz:

```
amqsbcg MQTTExampleQueue queue manager name
```

Příklad kódu

Soubor PubSync.java je úplný výpis kódu popsaného v tématu Procedura. Upravte třídu `Example` v produktu Obrázek 89 na stránce 465 , chcete-li potlačit výchozí parametry použité v souboru PubSync.java.

```
package com.ibm.mq.id;
import com.ibm.micro.client.mqttv3.*;
public class PubSync {
  public static void main(String[] args) {
    try {
      MqttClient client = new MqttClient(Example.TCPAddress, Example.clientId);
      MqttTopic topic = client.getTopic(Example.topicString);
      MqttMessage message = new MqttMessage(Example.publication.getBytes());
      message.setQos(Example.QoS);
      client.connect();
      System.out.println("Waiting for up to " + Example.sleepTimeout / 1000
          + " seconds for publication of \"" + message.toString()
          + "\" with QoS = " + message.getQos());
      System.out.println("On topic \"" + topic.getName()
          + "\" for client instance: \"" + client.getClientId()
          + "\" on address " + client.getServerURI() + "\"");
      MqttDeliveryToken token = topic.publish(message);
      token.waitForCompletion(Example.sleepTimeout);
      System.out.println("Delivery token \"" + token.hashCode()
          + "\" has been received: " + token.isComplete());
      client.disconnect();
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

Obrázek 88. PubSync.java

```
package com.ibm.mq.id;
import java.util.Properties;
import java.util.UUID;
public final class Example {
  public static final String          TCPAddress =
    System.getProperty("TCPAddress", "tcp://localhost:1883");
  public static final String          SSLAddress =
    System.getProperty("SSLAddress", "ssl://localhost:8883");
  public static final String            username =
    System.getProperty("username", System.getProperty("user.name"));
  public static final char []           password =
    System.getProperty("password", "Password").toCharArray();
  public static String                  clientId =
  String.format("%-23.23s", username  + "_" +
    System.getProperty("clientId",
    (UUID.randomUUID().toString())).trim()).replace('-', '_');
  public static final String         topicString =
    System.getProperty("topicString", "MQTT Example");
  public static final String         publication =
    System.getProperty("publication", "Hello World " +
  String.format("%tc", System.currentTimeMillis()));
  public static final int         quiesceTimeout =
    Integer.parseInt(System.getProperty("timeout", "10000"));
  public static final int           sleepTimeout =
    Integer.parseInt(System.getProperty("timeout", "10000"));
  public static final boolean       cleanSession =
    Boolean.parseBoolean(System.getProperty("cleanSession", "false"));
  public static final int                   QoS =
    Integer.parseInt(System.getProperty("QoS", "1"));
  public static final boolean          retained =
    Boolean.parseBoolean(System.getProperty("retained", "false"));
  public static final Properties getSSLSettings() {
    final Properties properties = new Properties();
    properties.setProperty("com.ibm.ssl.keyStore",
        "C:\\IBM\\MQ\\Data\\ClientKeyStore.jks");
    properties.setProperty("com.ibm.ssl.keyStoreType",
        "JKS");
    properties.setProperty("com.ibm.ssl.keyStorePassword",
        "password");
    properties.setProperty("com.ibm.ssl.trustStore",
        "C:\\IBM\\MQ\\Data\\ClientTrustStore.jks");
    properties.setProperty("com.ibm.ssl.trustStoreType",
        "JKS");
    properties.setProperty("com.ibm.ssl.trustStorePassword",
        "password");
    return properties;
  }
}
```

Obrázek 89. Example.java

Související pojmy

Aplikace typu publikování/odběr protokolu MQTT

Vytvoření asynchronního vydavatele pro přenos MQ Telemetry pomocí jazyka Java

V této úloze následujete výukový program k úpravě vaší první aplikace vydavatele. Úpravy umožňují aplikaci odesílat publikace bez čekání na potvrzení doručení. Potvrzení doručení jsou přijata třídou zpětného volání, kterou vytvoříte.

Než začnete

Nainstalujte funkci WebSphere MQ Telemetry na server, který má nainstalován produkt IBM WebSphere MQ Version 7.1 nebo novější.

Klientská aplikace používá balík produktu `com.ibm.mq.micro.client.mqttv3` v sadě nástrojů Software Development Toolkit (SDK) produktu IBM WebSphere MQ Telemetry . Sada SDK je součástí instalace produktu IBM WebSphere MQ Telemetry . Klient se připojuje k funkci IBM WebSphere MQ Telemetry pro výměnu zpráv s IBM WebSphere MQ.

Chcete-li spravovat produkt IBM WebSphere MQ Telemetry, musíte také nainstalovat aktualizace telemetrie pro produkt IBM WebSphere MQ Explorer Version 7.1 . Aktualizace jsou součástí instalace produktu IBM WebSphere MQ Telemetry .

Klient MQTT spuštěný v prostředí Java SE vyžaduje verzi 6.0 jazyka Java SE nebo novější. Produkt IBM Java SE v6.0 je součástí instalace produktu IBM WebSphere MQ Version 7.1 . Nachází se v *WebSphere MQ installation directory*\java\jre

Informace o této úloze

Příkladem je publikační aplikace, PubAsync. Produkt PubAsync publikuje `Hello World` na téma `MQTT Examples`, aniž by čekal na potvrzení, že publikování bylo doručeno správci front. Potvrzení o doručení jsou přijata ve třídě zpětného volání, `CallBack`.

Nastavením trvalého odběru na produkt `MQTT Examples` můžete zkontrolovat, zda tato aplikace pracuje.

Procedura používá platformu Eclipse k vývoji, sestavení a spuštění klienta. Eclipse si můžete stáhnout z webu projektu Eclipse na adrese www.eclipse.org.

Kroky v postupu upravují aplikaci PubSync`.java` v produktu “Vytvoření první aplikace MQ Telemetry Transport Publisher pomocí jazyka Java” na stránce 459.

Případně můžete zkopírovat kód “Příklad kódu” na stránce 468do nového adresáře `.\com\ibm\mq\id`. Vytvořte tři soubory Java, `Example.java`, `CallBack.javaa` PubAsync`.java`. Zkompilujte příklady pomocí příkazu,

```
javac -cp jar_dir\com.mq.micro.client.mqttv3.jar
       com.ibm.mq.id.PubAsync.java com.ibm.mq.id.CallBack.java com.ibm.mq.id.Example.java
```

Spusťte příkaz PubAsync pomocí příkazu,

```
java -cp jar_dir\com.mq.micro.client.mqttv3.jar
      com.ibm.mq.id.PubAsync.class
```

Postup

1. V balíku `com.ibm.mq.id` vytvořte soubor `CallBack.java`. Zkopírujte kód do souboru Obrázek 92 na stránce 468 do souboru.

   `Callback.java` implementuje rozhraní `MqttCallBack` . V tomto příkladu inicializuje další konstruktor zpětné volání s některými daty instance.
2. V balíku `com.ibm.mq.id` klepněte pravým tlačítkem myši na PubSync`.java` a zkopírujte jej. Vložte jej do stejného balíku, přejmenujte jej na PubAsync.
3. Těsně před řádkem kódu `client.connect();` , konkretizovat třídu `CallBack` , která předává identifikátor klienta.

   ```
   CallBack callback = new CallBack(Example.clientId);
   client.setCallback(callback);
   ```

   - Třída `CallBack` implementuje `MqttCallBack`. Pro identifikátor klienta je požadována jedna instance zpětného volání. V tomto příkladu konstruktor předává identifikátor klienta, který se má uložit jako data instance. Používá se ve zpětném volání k identifikaci instance zpětného volání, která byla spuštěna.
   - Ve třídě zpětného volání je třeba implementovat tři metody:

     `public void messageArrived(MqttTopic topic, MqttMessage message)`
     Přijímá publikování, k jehož odběru došlo k odběru.

     `public void connectionLost(Throwable cause)`
     Voláno při ztrátě připojení.

     `public void deliveryComplete(MqttDeliveryToken token))`
     Voláno, je-li přijat token doručení pro zprávu QoS 1 nebo 2, která byla publikována.

- Zpětné volání je aktivováno pomocí `MqttClient.connect`.

4. Odpojit klienta
   a) Odeberte příkaz obsahující výraz `token.waitForCompletion` .

      Hlavní podproces pokračuje bez čekání na doručení publikování.

   b) Otestujte, zda je klient již odpojen.

      Klient MQTT se odpojí po chybě, která byla vrácena metodě `lostConnection` v produktu `MqttCallback`, nebo se může klientská aplikace odpojit. Testujte, zda je k dispozici otevřené připojení.

   c) Chcete-li nastavit maximální dobu pro uvedení klienta do klidového stavu, použijte konstantu `Example.quiesceTimeout`.

```
if (client.isConnected())
    client.disconnect(Example.quiesceTimeout);
```

   Klient skončí, když je splněna kombinace následujících tří podmínek:

   a. Zpětné volání bylo voláno pro všechny zprávy, které byly publikovány v této relaci, nebo pokud byla relace restartována, v předchozích relacích.

   b. Zprávy jsou v letu a interval uvedení do klidového stavu vypršelo. Interval uvedení do klidového stavu je standardně 30 sekund. Časový limit uvedení do klidového stavu můžete změnit tak, že budete jako parametr produktu `client.disconnect`čekat, kolik milisekund bude čekat.

   c. Produkt `client.disconnect` byl zavolán poté, co byly některé zprávy publikovány a řazeny do fronty klientem, ale před odesláním zpráv. Zprávy ve frontě ještě nejsou v letu. Je-li relace restartovatelná, budou zprávy znovu odeslány, když se relace restartuje.

Výsledky

Chcete-li si prohlédnout publikace pomocí produktu WebSphere MQ, vytvořte téma, frontu a trvalý odběr, všechny nazývané `"MQTTExampleTopic"` používající skript v produktu Obrázek 90 na stránce 467. Spusťte klienta pro publikování na téma `MQTT Examples` a poté spusťte ukázkový program **amqsbcg** a procházejte publikace ve frontě produktu `MQTTExamples` .

1. Spusťte správce front a spusťte službu telemetrie (MQXR) spuštěnou. Ujistěte se, že adresa TCP/IP a port konfigurovaný pro kanál telemetrie se shodují s hodnotami, které používáte v aplikaci MQTT.
2. Nakonfigurujte trvalý odběr vytvořením příkazového skriptu `mqttexamples.txt` a jeho spuštěním s použitím produktu **runmqsc:** .

```
DEFINE TOPIC('MQTTExampleTopic') TOPICSTR('MQTT Example') REPLACE
DEFINE QLOCAL('MQTTExampleQueue') REPLACE
DEFINE SUB('MQTTExampleSub') DEST('MQTTExampleQueue') TOPICOBJ('MQTTExampleTopic') REPLACE
```

Obrázek 90. `mqttExampleTopic.txt`

   Chcete-li spustit skript v systému Windows, zadejte následující příkaz:

```
runmqsc queue manager name < mqttExampleTopic.txt
```

3. Spusťte klienta jako aplikaci Java z platformy Eclipse, nebo spuštěním jazyka Java v příkazovém okně:

```
java -cp jar_dir\com.mq.micro.client.mqttv3.jar
      com.ibm.mq.id.classname.class
```

   Poznámka: Příkazové okno musí být otevřeno v adresáři, který obsahuje cestu, `com\ibm\mq\id`.

4. Procházejte výsledky pomocí průzkumníka WebSphere MQ Explorer nebo spusťte příkaz:

```
amqsbcg MQTTExampleQueue queue manager name
```

Příklad kódu

```
package com.ibm.mq.id;
import com.ibm.micro.client.mqttv3.*;
public class PubAsync {
  public static void main(String[] args) {
    try {
      MqttClient client = new MqttClient(Example.TCPAddress, Example.clientId);
      MqttTopic topic = client.getTopic(Example.topicString);
      MqttMessage message = new MqttMessage(Example.publication.getBytes());
      CallBack callback = new CallBack(Example.clientId);
      client.setCallback(callback);
      client.connect();
      System.out.println("Publishing \"" + message.toString()
          + "\" on topic \"" + topic.getName() + "\" with QoS = "
          + message.getQos());
      System.out.println("For client instance \"" + client.getClientId()
          + "\" on address " + client.getServerURI() + "\"");
      MqttDeliveryToken token = topic.publish(message);
      System.out.println("With delivery token \"" + token.hashCode()
          + " delivered: " + token.isComplete());
      if (client.isConnected())
          client.disconnect(Example.quiesceTimeout);
      System.out.println("Disconnected: delivery token \"" + token.hashCode()
          + "\" received: " + token.isComplete());
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

Obrázek 91. PubAsync.java

```
package com.ibm.mq.id;
import com.ibm.micro.client.mqttv3.*;
public class CallBack implements MqttCallback {
  private String instanceData = "";
  public CallBack(String instance) {
    instanceData = instance;
  }
  public void messageArrived(MqttTopic topic, MqttMessage message) {
    try {
      System.out.println("Message arrived: \"" + message.toString()
          + "\" on topic \"" + topic.toString() + "\" for instance \""
          + instanceData + "\"");
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
  public void connectionLost(Throwable cause) {
    System.out.println("Connection lost on instance \"" + instanceData
        + "\" with cause \"" + cause.getMessage() + "\" Reason code "
        + ((MqttException)cause).getReasonCode() + "\" Cause \""
        + ((MqttException)cause).getCause() +  "\"");
    cause.printStackTrace();
  }
  public void deliveryComplete(MqttDeliveryToken token) {
    try {
      System.out.println("Delivery token \"" + token.hashCode()
          + "\" received by instance \"" + instanceData + "\"");
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

Obrázek 92. CallBack.java

```
package com.ibm.mq.id;
import java.util.Properties;
import java.util.UUID;
public final class Example {
  public static final String          TCPAddress =
    System.getProperty("TCPAddress", "tcp://localhost:1883");
  public static final String          SSLAddress =
    System.getProperty("SSLAddress", "ssl://localhost:8883");
  public static final String            username =
    System.getProperty("username", System.getProperty("user.name"));
  public static final char []           password =
    System.getProperty("password", "Password").toCharArray();
  public static String                  clientId =
  String.format("%-23.23s", username  + "_" +
    System.getProperty("clientId",
    (UUID.randomUUID().toString())).trim()).replace('-', '_');
  public static final String         topicString =
    System.getProperty("topicString", "MQTT Example");
  public static final String         publication =
    System.getProperty("publication", "Hello World " +
  String.format("%tc", System.currentTimeMillis()));
  public static final int         quiesceTimeout =
    Integer.parseInt(System.getProperty("timeout", "10000"));
  public static final int           sleepTimeout =
    Integer.parseInt(System.getProperty("timeout", "10000"));
  public static final boolean       cleanSession =
    Boolean.parseBoolean(System.getProperty("cleanSession", "false"));
  public static final int                   QoS =
    Integer.parseInt(System.getProperty("QoS", "1"));
  public static final boolean           retained =
    Boolean.parseBoolean(System.getProperty("retained", "false"));
  public static final Properties getSSLSettings() {
    final Properties properties = new Properties();
    properties.setProperty("com.ibm.ssl.keyStore",
        "C:\\IBM\\MQ\\Data\\ClientKeyStore.jks");
    properties.setProperty("com.ibm.ssl.keyStoreType",
        "JKS");
    properties.setProperty("com.ibm.ssl.keyStorePassword",
        "password");
    properties.setProperty("com.ibm.ssl.trustStore",
        "C:\\IBM\\MQ\\Data\\ClientTrustStore.jks");
    properties.setProperty("com.ibm.ssl.trustStoreType",
        "JKS");
    properties.setProperty("com.ibm.ssl.trustStorePassword",
        "password");
    return properties;
  }
}
```

Obrázek 93. Example.java

Vytvoření obnovitelného asynchronního vydavatele pro přenos MQ Telemetry pomocí jazyka Java

V této úloze budete postupovat podle výukového programu, abyste upravili asynchronní aplikaci vydavatele. Úpravy umožňují aplikaci dokončit doručení publikací, které nebyly potvrzeny při posledním spuštění klienta.

Než začnete

Nainstalujte funkci WebSphere MQ Telemetry na server, který má nainstalován produkt IBM WebSphere MQ Version 7.1 nebo novější.

Klientská aplikace používá balík produktu `com.ibm.mq.micro.client.mqttv3` v sadě nástrojů Software Development Toolkit (SDK) produktu IBM WebSphere MQ Telemetry . Sada SDK je součástí instalace produktu IBM WebSphere MQ Telemetry . Klient se připojuje k funkci IBM WebSphere MQ Telemetry pro výměnu zpráv s IBM WebSphere MQ.

Chcete-li spravovat produkt IBM WebSphere MQ Telemetry, musíte také nainstalovat aktualizace telemetrie pro produkt IBM WebSphere MQ Explorer Version 7.1 . Aktualizace jsou součástí instalace produktu IBM WebSphere MQ Telemetry .

Klient MQTT spuštěný v prostředí Java SE vyžaduje verzi 6.0 jazyka Java SE nebo novější. Produkt IBM Java SE v6.0 je součástí instalace produktu IBM WebSphere MQ Version 7.1 . Nachází se v *WebSphere MQ installation directory*\java\jre

Informace o této úloze

Příkladem je publikační aplikace, PubAsyncRestartable. Produkt PubAsyncRestartable publikuje dokument Hello World na téma MQTT Examples, aniž by čekal na potvrzení, že publikování bylo doručeno správci front. Potvrzení o doručení jsou přijata ve třídě zpětného volání, CallBack. Všechny tokeny doručení pro publikace, které nebyly dokončeny v předchozí instanci, lze vyšetřit. Jsou také zpracovány třídou zpětného volání.

Nastavením trvalého odběru na produkt MQTT Examples můžete zkontrolovat, zda tato aplikace pracuje.

Procedura používá platformu Eclipse k vývoji, sestavení a spuštění klienta. Eclipse si můžete stáhnout z webu projektu Eclipse na adrese www.eclipse.org.

Kroky v postupu upravují aplikaci PubAsync.java v produktu “Vytvoření asynchronního vydavatele pro přenos MQ Telemetry pomocí jazyka Java” na stránce 465.

Případně můžete zkopírovat kód “Příklad kódu” na stránce 473do nového adresáře .\com\ibm\mq\id. Vytvořte tři soubory Java, Example.java, CallBack.javaa PubAsyncRestartable.java. Zkompilujte příklady pomocí příkazu,

```
javac -cp jar_dir\com.mq.micro.client.mqttv3.jar
       com.ibm.mq.id.PubAsyncRestartable.java com.ibm.mq.id.CallBack.java
com.ibm.mq.id.Example.java
```

Spusťte příkaz PubAsyncRestartable pomocí příkazu,

```
java -cp jar_dir\com.mq.micro.client.mqttv3.jar
      com.ibm.mq.id.PubAsyncRestartable.class
```

Postup

1. V balíku com.ibm.mq.id klepněte pravým tlačítkem myši na PubAsync.java a zkopírujte jej. Vložte jej do stejného balíku, přejmenujte jej na PubAsyncRestartable.
2. Vytvořte opakovaně použitelný identifikátor klienta.

   ```
   Example.clientId = String.format(
       "%-23.23s",
       (System.getProperty("user.name") + "_" + (System.getProperty(
           "clientId", "PubAsyncRestartable."))).trim()).replace('-', '_');
   ```

   Obrázek 94. Znovu použitelný identifikátor klienta

   Aplikace v produktu “Vytvoření první aplikace MQ Telemetry Transport Publisher pomocí jazyka Java” na stránce 459 a “Vytvoření asynchronního vydavatele pro přenos MQ Telemetry pomocí jazyka Java” na stránce 465 používaly nový identifikátor klienta pro každé připojení klienta. Pro znovuspustitelného vydavatele nebo odběratele musíte použít stejný identifikátor klienta pokaždé, když je klient připojen, ale různí klienti musí používat různé identifikátory; viz “Identifikátor klienta” na stránce 512. Znovupoužitelný identifikátor klienta je sestaven ze jména uživatele a názvu třídy. Jeho délka je omezena na 23 bajtů. Musí mít pouze znaky, které jsou platné v názvech objektů správce front. Kód odstraní všechny spojovníky, které mohly být vloženy.
3. Hodnota QoS zprávy je nastavena na hodnotu 2, nikoli jako výchozí hodnota 1, aby nedošlo k duplicitnímu zprávám.

   ```
   message.setQos(Example.QoS);
   ```

Musíte buď změnit hodnotu `Example.QoS` na 2, nebo předat vlastnost `QoS` jako argument pomocí volby `-DQoS=2` na příkazovém řádku Java.

4. Vytvořte objekt `MqttConnectOptions` a nastavte jeho atribut `cleanSession` na hodnotu `false`.
   a) Vytvořte objekt `MqttConnectOptions` .

   ```
   MqttConnectOptions conOptions = new MqttConnectOptions();
   ```

   `conOptions` je parametr volby v konstruktoru `MqttClient` .
   b) Nastavte atribut clearSession .

   ```
   conOptions.setCleanSession(Example.cleanSession);
   ```

   Standardně je parametr `Example.cleanSession` nastaven na hodnotu `true`a odpovídá výchozímu nastavení parametru `MqttConnectionOptions.cleanSession`.

   Když se produkt `PubAsyncRestartable` restartuje, může začít s "čistou relací" a vymazat všechny čekající tokeny doručení pro zprávy QoS 1 nebo 2.

   Chcete-li zachovat všechny nevyřízené tokeny doručení, nastavte parametr `Example.cleanSession` na hodnotu `false` . Tokeny jsou zpracovány třídou `MqttCallBack` , když je klient opět připojen.
5. Je-li relace restartována, pak načtěte všechny nevyřízené doručovací tokeny a vytiskněte jejich obsah.

```
if (!conOptions.isCleanSession()) {
        MqttDeliveryToken tokens[] = client.getPendingDeliveryTokens();
        System.out.println("Starting a previous session for instance \""
            + client.getClientId() + "\" with " + tokens.length
            + " delivery tokens pending");
        for (int i = 0; i < tokens.length; i++) {
          System.out.println("Message \"" + tokens[i].getMessage().toString()
              + "\" with QoS=" + tokens[i].getMessage().getQos()
              + " recovered by instance \"" + client.getClientId()
              + "\" and assigned delivery token \"" + tokens[i].hashCode()
              + "\"");
        }
      } else
        System.out.println("Starting a clean session for instance "
            + client.getClientId());
```

6. Předejte parametr `conOptions` konstruktoru `MqttClient` .

```
client.connect(conOptions);
```

7. Při odpojování nastavte maximální interval odpojení.

```
client.disconnect(Example.timeout);
```

Aby bylo možné zobrazit nevyřízené tokeny doručení, musí předchozí instance skončit bez dokončení doručení. Chcete-li spustit tento příklad s možností nepotvrzení publikování před dokončením `PubAsyncRestartable` , nastavte `Example.timeout` na 0.

Výsledky

Chcete-li si prohlédnout publikace pomocí produktu WebSphere MQ, vytvořte téma, frontu a trvalý odběr, všechny nazývané `"MQTTExampleTopic"` používající skript v produktu Obrázek 95 na stránce 472. Spusťte klienta pro publikování na téma `MQTT Examples` a poté spusťte ukázkový program **amqsbcg** a procházejte publikace ve frontě produktu `MQTTExamples` .

1. Spusťte správce front a spusťte službu telemetrie (MQXR) spuštěnou. Ujistěte se, že adresa TCP/IP a port konfigurovaný pro kanál telemetrie se shodují s hodnotami, které používáte v aplikaci MQTT.
2. Nakonfigurujte trvalý odběr vytvořením příkazového skriptu `mqttexamples.txt` a jeho spuštěním s použitím produktu **runmqsc:** .

```
DEFINE TOPIC('MQTTExampleTopic') TOPICSTR('MQTT Example') REPLACE
DEFINE QLOCAL('MQTTExampleQueue') REPLACE
DEFINE SUB('MQTTExampleSub') DEST('MQTTExampleQueue') TOPICOBJ('MQTTExampleTopic') REPLACE
```

Obrázek 95. mqttExampleTopic.txt

Chcete-li spustit skript v systému Windows, zadejte následující příkaz:

```
runmqsc queue manager name < mqttExampleTopic.txt
```

3. Spusťte klienta jako aplikaci Java z platformy Eclipse, nebo spuštěním jazyka Java v příkazovém okně:

```
java -cp jar_dir\com.mq.micro.client.mqttv3.jar
      com.ibm.mq.id.classname.class
```

Poznámka: Příkazové okno musí být otevřeno v adresáři, který obsahuje cestu, `com\ibm\mq\id`.

4. Procházejte výsledky pomocí průzkumníka WebSphere MQ Explorer nebo spusťte příkaz:

```
amqsbcg MQTTExampleQueue queue manager name
```

Příklad kódu

```
package com.ibm.mq.id;
import com.ibm.micro.client.mqttv3.MqttClient;
import com.ibm.micro.client.mqttv3.MqttConnectOptions;
import com.ibm.micro.client.mqttv3.MqttDeliveryToken;
import com.ibm.micro.client.mqttv3.MqttMessage;
import com.ibm.micro.client.mqttv3.MqttTopic;
public class PubAsyncRestartable {
  public static void main(String[] args) {
    Example.clientId = String.format(
        "%-23.23s",
        (System.getProperty("user.name") + "_" + (System.getProperty(
            "clientId", "PubAsyncRestartable."))).trim()).replace('-', '_');
    try {
      MqttClient client = new MqttClient(Example.TCPAddress, Example.clientId);
      MqttTopic topic = client.getTopic(Example.topicString);
      MqttMessage message = new MqttMessage(Example.publication.getBytes());
      message.setQos(Example.QoS);
      CallBack callback = new CallBack(Example.clientId);
      client.setCallback(callback);
      MqttConnectOptions conOptions = new MqttConnectOptions();
      conOptions.setCleanSession(Example.cleanSession);
      if (!conOptions.isCleanSession()) {
        MqttDeliveryToken tokens[] = client.getPendingDeliveryTokens();
        System.out.println("Starting a previous session for instance \""
            + client.getClientId() + "\" with " + tokens.length
            + " delivery tokens pending");
        for (int i = 0; i < tokens.length; i++) {
          System.out.println("Message \"" + tokens[i].getMessage().toString()
              + "\" with QoS=" + tokens[i].getMessage().getQos()
              + " recovered by instance \"" + client.getClientId()
              + "\" and assigned delivery token \"" + tokens[i].hashCode()
              + "\"");
        }
      } else
        System.out.println("Starting a clean session for instance \""
            + client.getClientId() + "\"");
      client.connect(conOptions);
      System.out.println("Publishing \"" + message.toString()
          + "\" on topic \"" + topic.getName() + "\" with QoS = "
          + message.getQos());
      System.out.println("For client instance \"" + client.getClientId()
          + "\" on address " + client.getServerURI() + "\"");
      MqttDeliveryToken token = topic.publish(message);
      System.out.println("With delivery token \"" + token.hashCode()
          + " delivered: " + token.isComplete());
      if (client.isConnected())
          client.disconnect(Example.quiesceTimeout);
      System.out.println("Disconnected: delivery token \"" + token.hashCode()
          + "\" has been received: " + token.isComplete());
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

Obrázek 96. PubAsyncRestartable.java

```
package com.ibm.mq.id;
import com.ibm.micro.client.mqttv3.*;
public class CallBack implements MqttCallback {
  private String instanceData = "";
  public CallBack(String instance) {
    instanceData = instance;
  }
  public void messageArrived(MqttTopic topic, MqttMessage message) {
    try {
      System.out.println("Message arrived: \"" + message.toString()
          + "\" on topic \"" + topic.toString() + "\" for instance \""
          + instanceData + "\"");
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
  public void connectionLost(Throwable cause) {
    System.out.println("Connection lost on instance \"" + instanceData
        + "\" with cause \"" + cause.getMessage() + "\" Reason code "
        + ((MqttException)cause).getReasonCode() + "\" Cause \""
        + ((MqttException)cause).getCause() +  "\"");
    cause.printStackTrace();
  }
  public void deliveryComplete(MqttDeliveryToken token) {
    try {
      System.out.println("Delivery token \"" + token.hashCode()
          + "\" received by instance \"" + instanceData + "\"");
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

Obrázek 97. CallBack.java

```
package com.ibm.mq.id;
import java.util.Properties;
import java.util.UUID;
public final class Example {
  public static final String          TCPAddress =
    System.getProperty("TCPAddress", "tcp://localhost:1883");
  public static final String          SSLAddress =
    System.getProperty("SSLAddress", "ssl://localhost:8883");
  public static final String            username =
    System.getProperty("username", System.getProperty("user.name"));
  public static final char []           password =
    System.getProperty("password", "Password").toCharArray();
  public static String                  clientId =
  String.format("%-23.23s", username  + "_" +
    System.getProperty("clientId",
    (UUID.randomUUID().toString())).trim()).replace('-', '_');
  public static final String         topicString =
    System.getProperty("topicString", "MQTT Example");
  public static final String         publication =
    System.getProperty("publication", "Hello World " +
  String.format("%tc", System.currentTimeMillis()));
  public static final int         quiesceTimeout =
    Integer.parseInt(System.getProperty("timeout", "10000"));
  public static final int           sleepTimeout =
    Integer.parseInt(System.getProperty("timeout", "10000"));
  public static final boolean       cleanSession =
    Boolean.parseBoolean(System.getProperty("cleanSession", "false"));
  public static final int                   QoS =
    Integer.parseInt(System.getProperty("QoS", "1"));
  public static final boolean          retained =
    Boolean.parseBoolean(System.getProperty("retained", "false"));
  public static final Properties getSSLSettings() {
    final Properties properties = new Properties();
    properties.setProperty("com.ibm.ssl.keyStore",
        "C:\\IBM\\MQ\\Data\\ClientKeyStore.jks");
    properties.setProperty("com.ibm.ssl.keyStoreType",
        "JKS");
    properties.setProperty("com.ibm.ssl.keyStorePassword",
        "password");
    properties.setProperty("com.ibm.ssl.trustStore",
        "C:\\IBM\\MQ\\Data\\ClientTrustStore.jks");
    properties.setProperty("com.ibm.ssl.trustStoreType",
        "JKS");
    properties.setProperty("com.ibm.ssl.trustStorePassword",
        "password");
    return properties;
  }
}
```

Obrázek 98. Example.java

Vytvoření odběratele pro přenos MQ Telemetry pomocí jazyka Java

V této úloze následujete výukový program k vytvoření aplikace odběratele. Odběratel vytvoří odběr na téma a obdrží publikování pro daný odběr.

Než začnete

Nainstalujte funkci WebSphere MQ Telemetry na server, který má nainstalován produkt IBM WebSphere MQ Version 7.1 nebo novější.

Klientská aplikace používá balík produktu `com.ibm.mq.micro.client.mqttv3` v sadě nástrojů Software Development Toolkit (SDK) produktu IBM WebSphere MQ Telemetry . Sada SDK je součástí instalace produktu IBM WebSphere MQ Telemetry . Klient se připojuje k funkci IBM WebSphere MQ Telemetry pro výměnu zpráv s IBM WebSphere MQ.

Chcete-li spravovat produkt IBM WebSphere MQ Telemetry, musíte také nainstalovat aktualizace telemetrie pro produkt IBM WebSphere MQ Explorer Version 7.1 . Aktualizace jsou součástí instalace produktu IBM WebSphere MQ Telemetry .

Klient MQTT spuštěný v prostředí Java SE vyžaduje verzi 6.0 jazyka Java SE nebo novější. Produkt IBM Java SE v6.0 je součástí instalace produktu IBM WebSphere MQ Version 7.1 . Nachází se v *WebSphere MQ installation directory*\java\jre

Informace o této úloze

Příkladem je aplikace odběratele, Subscribe. Produkt Subscribe vytvoří téma odběru MQTT Examplesa čeká na publikování v odběru po dobu 30 sekund.

Odběratel může vytvořit odběr a čekat na publikování. Může také obdržet publikace odeslané do dříve vytvořeného odběru pro stejný identifikátor klienta. Logický atribut MqttConnectionOptions.cleanSession řídí, zda byly dříve odeslané publikování přijaty, nebo ne; viz “Odběry” na stránce 523.

Publikační programy můžete použít k vytvoření publikování nebo k vytvoření testovacího publikování v tématu MQTT Examples pomocí produktu WebSphere MQ Explorer.

Procedura používá platformu Eclipse k vývoji, sestavení a spuštění klienta. Eclipse si můžete stáhnout z webu projektu Eclipse na adrese www.eclipse.org.

Pokyny v části Procedura předpokládají, že jste již vytvořili balík produktu com.ibm.mq.id v jedné z předchozích úloh a zkopírovali jste do tříd Example.java a Callback.java .

Postup

1. Vytvořte třídu Subscribe v balíku com.ibm.mq.id.
2. Vytvořte opakovaně použitelný identifikátor klienta.

```
Example.clientId = String.format(
    "%-23.23s",
    (System.getProperty("user.name") + "_" + (System.getProperty(
        "clientId", "Subscribe."))).trim()).replace('-', '_');
```

Obrázek 99. Znovu použitelný identifikátor klienta

Aplikace v produktu “Vytvoření první aplikace MQ Telemetry Transport Publisher pomocí jazyka Java” na stránce 459 a “Vytvoření asynchronního vydavatele pro přenos MQ Telemetry pomocí jazyka Java” na stránce 465 používaly nový identifikátor klienta pro každé připojení klienta. Pro znovuspustitelného vydavatele nebo odběratele musíte použít stejný identifikátor klienta pokaždé, když je klient připojen, ale různí klienti musí používat různé identifikátory; viz “Identifikátor klienta” na stránce 512. Znovupoužitelný identifikátor klienta je sestaven ze jména uživatele a názvu třídy. Jeho délka je omezena na 23 bajtů. Musí mít pouze znaky, které jsou platné v názvech objektů správce front. Kód odstraní všechny spojovníky, které mohly být vloženy.

3. Vytvořte blok try-catch .

```
try { ...
 } catch (Exception e) {
  e.printStackTrace();
}
```

Klient MQTT generuje MqttException, MqttPersistenceException nebo MqttSecurityException. MqttPersistenceException a MqttSecurityException jsou podtřídy MqttException.

Chcete-li zjistit příčinu výjimky, použijte metodu produktu MqttException.getReasonCode . Je-li MqttPersistenceException nebo MqttSecurityException hozena, použijte metodu getCause k vrácení podkladové události typu throwable.

4. Vytvoření nové instance MqttClient .

```
MqttClient client = new MqttClient(Example.TCPAddress, Example.clientId);
```

Zadejte klienta s adresou serveru, která se používá později k připojení k produktu WebSphere MQ. Nastavte identifikátor klienta tak, aby pojmenujete klienta.

- Volitelně můžete poskytnout implementaci rozhraní produktu `MqttClientPersistence` , která nahradí výchozí implementaci. Výchozí implementace produktu `MqttPersistence` ukládá QoS 1 a 2 zprávy čekající na doručení jako soubory; viz “Perzistence zpráv v klientech produktu MQTT” na stránce 516.
- The default IBM WebSphere MQ TCP/IP port for MQTT is 1883. Pro zabezpečení SSL je to 8883. V tomto příkladu je výchozí adresa nastavena na `tcp://localhost:1883`.
- Obvykle je důležité, abyste mohli identifikovat specifického fyzického klienta pomocí identifikátoru klienta. Identifikátor klienta musí být jedinečný v rámci všech klientů, kteří se připojují k serveru; viz “Identifikátor klienta” na stránce 512. Použití stejného identifikátoru klienta jako předchozí instance označuje, že aktuální instance je instancí stejného klienta. Pokud duplikujete identifikátor klienta ve dvou spuštěných klientech, dojde k výjimce v obou klientech a jeden z klientů se ukončí.
- Délka identifikátoru klienta je omezena na 23 bajtů. Dojde-li k překročení délky, dojde k výjimce. Identifikátor klienta musí obsahovat pouze znaky povolené v názvu správce front, například bez pomlček nebo mezer.
- Dokud nezavoláte metodu `MqttClient.connect` , žádné zpracování zpráv se neprovádí.

Pomocí objektu klienta můžete publikovat a odebírat témata a obnovovat informace o publikatech, které dosud nebyly doručeny.

5. Těsně před řádkem kódu `client.connect();` , konkretizovat třídu `CallBack` , která předává identifikátor klienta.

```
CallBack callback = new CallBack(Example.clientId);
client.setCallback(callback);
```

- Třída `CallBack` implementuje `MqttCallBack`. Pro identifikátor klienta je požadována jedna instance zpětného volání. V tomto příkladu konstruktor předává identifikátor klienta, který se má uložit jako data instance. Používá se ve zpětném volání k identifikaci instance zpětného volání, která byla spuštěna.
- Ve třídě zpětného volání je třeba implementovat tři metody:

  `public void messageArrived(MqttTopic topic, MqttMessage message)`
  Přijímá publikování, k jehož odběru došlo k odběru.

  `public void connectionLost(Throwable cause)`
  Voláno při ztrátě připojení.

  `public void deliveryComplete(MqttDeliveryToken token))`
  Voláno, je-li přijat token doručení pro zprávu QoS 1 nebo 2, která byla publikována.
- Zpětné volání je aktivováno pomocí `MqttClient.connect`.

6. Vytvořte objekt `MqttConnectOptions` a nastavte jeho atribut `cleanSession` .
   a) Vytvořte objekt `MqttConnectOptions` .

```
MqttConnectOptions conOptions = new MqttConnectOptions();
```

   `conOptions` je parametr volby v konstruktoru `MqttClient` .

   b) Nastavte atribut clearSession .

```
conOptions.setCleanSession(Example.cleanSession);
```

   Standardně je parametr `Example.cleanSession` nastaven na hodnotu `true`a odpovídá výchozímu nastavení parametru `MqttConnectionOptions.cleanSession`.

Používáte-li výchozí hodnotu `MqttConnectOptions` nebo před připojením klienta nastavíte `MqttConnectOptions.cleanSession` na hodnotu `true`, budou všechny staré odběry klienta při připojení odebrány. Všechny nové odběry, které klient během relace vytvoří, jsou při odpojení odebrány.

Pokud před připojením nastavíte `MqttConnectOptions.cleanSession` na hodnotu `false`, budou všechny vytvořené odběry klienta přidány ke všem odběrům, které existovaly pro klienta 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 je třeba nastavit režim `cleanSession`; režim trvá pro celou relaci. Chcete-li změnit jeho nastavení, je třeba klienta odpojit a znovu připojit. Pokud změníte režimy z používání `cleanSession=false` na `cleanSession=true`, budou všechny předchozí odběry pro klienta a všechny publikace, které nebyly přijaty, vyřazeny.

7. Předejte parametr `conOptions` konstruktoru `MqttClient` .

```
client.connect(conOptions);
```

8. Vytvořte odběr.

```
      client.subscribe(Example.topicString, Example.QoS);
```

Tento příklad používá metodu `MqttClient.subscribe` , která předává jeden filtr tématu s volbou `QoS` . Metoda `MqttClient.subscribe` má čtyři podpisy a můžete také předat pole filtrů odběrů a také jeden filtr.

Tento příklad používá řetězec tématu používaný příklady publikování jako filtr témat, takže přijímá jakékoli publikace, které vytvoří.

Při každém spuštění tohoto příkladu produkt `subscribe.java`vytvoří odběr. Pokud Example.topicStringnezměníte, znovu vytvoří stejný odběr znovu. Je-li odběr znovu vytvořen, nebude mít za následek dva identické odběry. Klient neobdržel duplicitní kopie publikací, které se shodují s identickým odběrem.

Odběry jsou popsány v tématu “Odběry” na stránce 523a filtry v produktu “Řetězce témat a filtry témat v klientech MQTT” na stránce 525.

9. Počkejte, až některé publikace dorazí a pak odpojí klienta.

```
      Thread.sleep(Example.sleepTimeout);
      client.disconnect();
```

Publikace jsou přijímány implementací metody `MqttCallback.messageArrived` .

Aplikace odběru nepublikovala žádné zprávy, a tak nečeká na žádné tokeny doručení. `client.disconnect` se provádí bez prodlevy.

Příklad kódu

```
package com.ibm.mq.id;
import com.ibm.micro.client.mqttv3.MqttClient;
import com.ibm.micro.client.mqttv3.MqttConnectOptions;
public class Subscribe {
  public static void main(String[] args) {
    Example.clientId = String.format(
        "%-23.23s",
        (System.getProperty("user.name") + "_" + System.getProperty("clientId",
            "Subscribe."))).trim();
    try {
      MqttClient client = new MqttClient(Example.TCPAddress, Example.clientId);
      CallBack callback = new CallBack(Example.clientId);
      client.setCallback(callback);
      MqttConnectOptions conOptions = new MqttConnectOptions();
      conOptions.setCleanSession(Example.cleanSession);
      client.connect(conOptions);
      System.out.println("Subscribing to topic \"" + Example.topicString
          + "\" for client instance \"" + client.getClientId()
          + "\" using QoS " + Example.QoS + ". Clean session is "
          + Example.cleanSession);
      client.subscribe(Example.topicString, Example.QoS);
      System.out.println("Going to sleep for " + Example.sleepTimeout / 1000
          + " seconds");
      Thread.sleep(Example.sleepTimeout);
      client.disconnect();
      System.out.println("Finished");
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

Obrázek 100. Subscribe.java

```
package com.ibm.mq.id;
import com.ibm.micro.client.mqttv3.*;
public class CallBack implements MqttCallback {
  private String instanceData = "";
  public CallBack(String instance) {
    instanceData = instance;
  }
  public void messageArrived(MqttTopic topic, MqttMessage message) {
    try {
      System.out.println("Message arrived: \"" + message.toString()
          + "\" on topic \"" + topic.toString() + "\" for instance \""
          + instanceData + "\"");
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
  public void connectionLost(Throwable cause) {
    System.out.println("Connection lost on instance \"" + instanceData
        + "\" with cause \"" + cause.getMessage() + "\" Reason code "
        + ((MqttException)cause).getReasonCode() + "\" Cause \""
        + ((MqttException)cause).getCause() +  "\"");
    cause.printStackTrace();
  }
  public void deliveryComplete(MqttDeliveryToken token) {
    try {
      System.out.println("Delivery token \"" + token.hashCode()
          + "\" received by instance \"" + instanceData + "\"");
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

Obrázek 101. CallBack.java

```
package com.ibm.mq.id;
import java.util.Properties;
import java.util.UUID;
public final class Example {
  public static final String          TCPAddress =
    System.getProperty("TCPAddress", "tcp://localhost:1883");
  public static final String          SSLAddress =
    System.getProperty("SSLAddress", "ssl://localhost:8883");
  public static final String            username =
    System.getProperty("username", System.getProperty("user.name"));
  public static final char []           password =
    System.getProperty("password", "Password").toCharArray();
  public static String                  clientId =
  String.format("%-23.23s", username  + "_" +
    System.getProperty("clientId",
    (UUID.randomUUID().toString())).trim()).replace('-', '_');
  public static final String         topicString =
    System.getProperty("topicString", "MQTT Example");
  public static final String         publication =
    System.getProperty("publication", "Hello World " +
  String.format("%tc", System.currentTimeMillis()));
  public static final int         quiesceTimeout =
    Integer.parseInt(System.getProperty("timeout", "10000"));
  public static final int           sleepTimeout =
    Integer.parseInt(System.getProperty("timeout", "10000"));
  public static final boolean       cleanSession =
    Boolean.parseBoolean(System.getProperty("cleanSession", "false"));
  public static final int                   QoS =
    Integer.parseInt(System.getProperty("QoS", "1"));
  public static final boolean           retained =
    Boolean.parseBoolean(System.getProperty("retained", "false"));
  public static final Properties getSSLSettings() {
    final Properties properties = new Properties();
    properties.setProperty("com.ibm.ssl.keyStore",
        "C:\\IBM\\MQ\\Data\\ClientKeyStore.jks");
    properties.setProperty("com.ibm.ssl.keyStoreType",
        "JKS");
    properties.setProperty("com.ibm.ssl.keyStorePassword",
        "password");
    properties.setProperty("com.ibm.ssl.trustStore",
        "C:\\IBM\\MQ\\Data\\ClientTrustStore.jks");
    properties.setProperty("com.ibm.ssl.trustStoreType",
        "JKS");
    properties.setProperty("com.ibm.ssl.trustStorePassword",
        "password");
    return properties;
  }
}
```

Obrázek 102. Example.java

Související pojmy

Aplikace typu publikování/odběr protokolu MQTT

Ověřování klienta MQTT Java pomocí JAAS

Naučte se, jak ověřit klienta pomocí služby JAAS. Upravte vzorový program `JAASLoginModule.java` a vzorového programu v Javě `PubSync.java`. Konfigurujte kanál telemetrie tak, aby vyžadoval ověřování JAAS , a spusťte upravený vydavatel, zkontrolujte jeho `jméno uživatele` a `heslo` pomocí JAAS.

Než začnete

Předpokládá se, že jste nainstalovali soubory JAR klienta protokolu MQTT v3 , dokumentaci Javadoc, Eclipse, nakonfigurované kanály telemetrie a před provedením této úlohy zakódované a spuštěné PubSync.java . Máte pracovní prostor Eclipse , který obsahuje spuštěnou verzi souboru PubSync.java.

Úloha je zapsána pro systém Windows. Změňte cesty k adresářům pro produkt Linux.

Informace o této úloze

Úloha je založena na úpravě ukázkové třídy JAASLoginModule v produktu *WMQ Installation directory*\mqxr\samples\JAASLoginModule.java za účelem vytvoření MyLogin.java. V úloze také upravíte ukázkový kód PubSync.java v produktu "Vytvoření první aplikace MQ Telemetry Transport Publisher pomocí jazyka Java" na stránce 459 a nastaví username a password. Jako test MyLogin.java náhodně přijímá nebo odmítá jméno uživatele a heslo.

Kroky v úloze se zapisují jako programovací cvičení. Chcete-li provést skutečnou autentizaci v produkčním prostředí, musíte upravit proceduru.

V typickém vysvětlení způsobu ověřování programu JAAS se předpokládá, že přihlašovací modul ověřuje kontext, který načten JAAS. Když služba telemetrie (MQXR) volá službu JAAS, kontext, který načetl službu JAAS , je služba telemetrie (MQXR). Při ověřování kontextu služby telemetrie (MQXR) není k dispozici žádný bod; jedná se o adresář mqm. Místo toho, služba telemetrie (MQXR) nastaví klienta username a password , aby byl k dispozici pro třídu přihlašovacího modulu. Parametry username a password jsou předány do přihlašovacího modulu pomocí dvou zpětných volání.

```
javax.security.auth.callback.Callback[] callbacks =
      new javax.security.auth.callback.Callback[2];
callbacks[0] =
      new javax.security.auth.callback.NameCallback("NameCallback");
callbacks[1] =
      new javax.security.auth.callback.PasswordCallback("PasswordCallback", false);
callbackHandler.handle(callbacks);
String username =
      ((javax.security.auth.callback.NameCallback) callbacks[0]).getName();
char[] password =
      ((javax.security.auth.callback.PasswordCallback) callbacks[1]).getPassword();
```

Jméno uživatele a heslo klienta jsou jediné informace o klientovi, který je k dispozici pro přihlašovací modul.

Postup

1. Vytvořte dva balíky, samples a security.jaas ve stejném projektu Java jako PubSync.java.

   Balík samples se používá pouze pro odkaz. Proveďte změny kódu v balíku security.jaas .
2. Naimportujte JAASLoginModule.java a JAASPrincipal.java do obou balíků.

   Je-li to nutné, refaktorovat příkazy balíku ve zdroji Java, aby se vyloučily chyby kompilace.
3. Refaktorujte název třídy JAASLoginModulev balíku security.jaas na hodnotu MyLogin .
4. V produktu MyLogin.javanahraďte některý kód v metodě login , aby zobrazoval modul.
   a) Nahraďte kód:

   ```
   // Accept everything.
   if (true)
     loggedIn = true;
   else
     throw new javax.security.auth.login.FailedLoginException("Login failed");
   ```

   b) S kódem:

   ```
   // login half the users randomly
   PrintWriter pw = new PrintWriter(new FileWriter(System.getProperty("user.dir")
           + "\\MyLogin.log", true));
   pw.println("Called JAASLogin.login at "
           + System.getProperty("publication", "Hello World "
           + String.format("%tc", System.currentTimeMillis())));
   if (Math.random() < 0.5)
     loggedIn = true;
   pw.println("Username: \"" + username + "\", Password: \""
       + String.valueOf(password) + "\" loggedIn: " + loggedIn);
   pw.close();
   if (!loggedIn)
     throw new javax.security.auth.login.FailedLoginException("Login failed");
   principal= new JAASPrincipal(username);
   ```

Úplný zdroj pro produkt `MyLogin.java` je v produktu Obrázek 105 na stránce 484. Zdroj pro `JAASPrincipal.java`, s názvem balíku refaktorován na `security.jaas` je v Obrázek 106 na stránce 485.

5. Nastavte cestu ke třídě v `service.env` tak, aby ukazovala na adresář obsahující cestu k `security/jaas/MyLogin.class` a `security/jaas/JAASPrincipal.class`.

```
CLASSPATH=C:\WMQTelemetryApps\MQTTSecureExamples\bin
```

Informace o použití produktu `service.env` k předání cesty ke třídě do služby produktu WebSphere MQ naleznete v tématu Konfigurace kanálu JAAS kanálu telemetrie .

6. Přidejte sekci přihlašovacího modulu do `jaas.config`.

```
MyLoginExample {
  security.jaas.MyLogin required debug=true;
  };
```

Informace o použití produktu `jaas.config` definují přihlašovací modul služby JAAS naleznete v tématu Konfigurace kanálu JAAS kanálu telemetrie .

7. Přidejte kanál telemetrie pomocí průvodce **Nový kanál telemetrie** v produktu WebSphere MQ Explorer a konfiguraci kanálu tak, aby vyžadoval ověřování služby JAAS . Odkažte jej na stanzu `MyLoginExample` .

Můžete například upravit informace, které zadáte do průvodce, z této stanzy v souboru `mqxr_win.properties` . Pokud pracujete v produktu Linux , je tento soubor nazván `mqxr_unix.properties`. Neupravujte soubor vlastností telemetrie přímo; použijte průvodce.

```
com.ibm.mq.MQXR.channel/JAASMCAUser: \
com.ibm.mq.MQXR.Port=1884;\
com.ibm.mq.MQXR.JAASConfig=MyLoginExample;\
com.ibm.mq.MQXR.UserName=Admin;\
com.ibm.mq.MQXR.StartWithMQXRService=true
```

Poznámka: Pokud upravíte některý z parametrů kanálu telemetrie nebo upravíte třídu `security.jaas.Mylogin` , je třeba zastavit a restartovat službu telemetrie (MQXR). Pouze když restartujete službu, provedené změny se projeví.

8. Vytvořte kopii produktu `PubSync.java` v balíku `com.ibm.mq.id` a pojmenujte kopii `PubSyncJAAS.java`.

Postup vytvoření souboru PubSync.java v balíku produktu `com.ibm.mq.id` naleznete v tématu “Vytvoření první aplikace MQ Telemetry Transport Publisher pomocí jazyka Java” na stránce 459 .

9. Nastavte parametry `MqttConnectOptions.username` a `MqttConnectOptions.password` v programu `PubSyncJAAS.java` a předejte `MqttConnectOptions` jako parametr `MqttClient.connect`.

```
MqttConnectOptions conOptions = new MqttConnectOptions();
conOptions.setUserName(Example.username);
conOptions.setPassword(Example.password);
client.connect(conOptions);
```

Viz kurzívazovaný kód v adresáři PubSyncJAAS.Java pomocí konstant nastavených v souboru Example.java.

10. Nastavte volbu `Example.TCPAddress` na adresu soketu kanálu telemetrie, který jste nakonfigurovali pro použití konfigurace služby JAAS , `MyLoginExample`. Jako číslo portu použijte 1884 .
11. Spusťte `PubSyncJAAS` krát, abyste viděli přihlášení klienta a byli přijati nebo odmítnuti.

Při každém zamítnutí pokusu o přihlášení dojde k výjimce.

Výsledky

Příkaz Obrázek 103 na stránce 483 zobrazuje výsledky spuštění služby PubSyncJAAS.Java dvakrát. Záznamy protokolu jsou zobrazeny v Obrázek 104 na stránce 483.

```
Waiting for up to 10 seconds for publication of "Hello World Fri Jun 04 08:31:05 BST 2010" with
QoS = 1
On topic "MQTT Example" for client instance: "Admin_61c57a18_4bf7_40d" on address tcp://
localhost:1884"
With username "Admin" and password "Password"
Client exception caught
Client is not connected (32104)
    at
com.ibm.micro.client.mqttv3.internal.ExceptionHelper.createMqttException(ExceptionHelper.java:33
)
    at com.ibm.micro.client.mqttv3.internal.ClientComms.internalSend(ClientComms.java:88)
    at com.ibm.micro.client.mqttv3.internal.ClientComms.sendNoWait(ClientComms.java:105)
    at com.ibm.micro.client.mqttv3.MqttTopic.publish(MqttTopic.java:68)
    at com.ibm.mq.id.PubSync.main(PubSync.java:24)
```

```
Waiting for up to 10 seconds for publication of "Hello World Fri Jun 04 08:31:40 BST 2010" with
QoS = 1
On topic "MQTT Example" for client instance: "Admin_1d1599a0_50f5_4ea" on address tcp://
localhost:1884"
With username "Admin" and password "Password"
Delivery token "1731749688" has been received: true
```

Obrázek 103. Výstup konzoly z PubSyncJAAS.java

Soubor protokolu MyLogin.log je uložen v adresáři *WMQ Data directory*; například: C:\IBM\MQ\Data\MyLogin.log:

```
Called JAASLogin.login at Hello World Fri Jun 04 08:31:05 BST 2010
Username: "Admin", Password: "Password" loggedIn: false
Called JAASLogin.login at Hello World Fri Jun 04 08:31:40 BST 2010
Username: "Admin", Password: "Password" loggedIn: true
```

Obrázek 104. MyLogin.log

Příklady

Položka s kurzívou v produktu Obrázek 105 na stránce 484 je úprava ukázky JAASLoginModule.java.

```
package security.jaas;
import java.io.FileWriter;
import java.io.PrintWriter;

public class JAASLogin implements javax.security.auth.spi.LoginModule {
  private javax.security.auth.Subject                  subject;
  private javax.security.auth.callback.CallbackHandler callbackHandler;
  JAASPrincipal                                        principal;
  boolean                                              loggedIn = false;
  public void initialize(javax.security.auth.Subject subject,
      javax.security.auth.callback.CallbackHandler callbackHandler,
      java.util.Map<String, ?> sharedState, java.util.Map<String, ?> options) {
    this.subject = subject;
    this.callbackHandler = callbackHandler;
  }
  public boolean login() throws javax.security.auth.login.LoginException {
    try {
      javax.security.auth.callback.Callback[] callbacks = new
javax.security.auth.callback.Callback[2];
      callbacks[0] = new javax.security.auth.callback.NameCallback(
          "NameCallback");
      callbacks[1] = new javax.security.auth.callback.PasswordCallback(
          "PasswordCallback", false);

      callbackHandler.handle(callbacks);
      String username = ((javax.security.auth.callback.NameCallback) callbacks[0])
          .getName();
      char[] password = ((javax.security.auth.callback.PasswordCallback) callbacks[1])
          .getPassword();
      // login half the users randomly
      PrintWriter pw = new PrintWriter(new FileWriter(System
          .getProperty("user.dir")
          + "\\mylogin.log", true));
      pw.println("Called JAASLogin.login at "
          + System.getProperty("publication", "Hello World "
              + String.format("%tc", System.currentTimeMillis())));
      if (Math.random() < 0.5)
        loggedIn = true;
      pw.println("Username: \"" + username + "\", Password: \""
          + String.valueOf(password) + "\" loggedIn: " + loggedIn);
      pw.close();
      if (!loggedIn)
        throw new javax.security.auth.login.FailedLoginException("Login failed");
      principal = new JAASPrincipal(username);
    } catch (java.io.IOException exception) {
      throw new javax.security.auth.login.LoginException(exception.toString());
    } catch (javax.security.auth.callback.UnsupportedCallbackException exception) {
      throw new javax.security.auth.login.LoginException(exception.toString());
    }
    return loggedIn;
  }
  public boolean abort() throws javax.security.auth.login.LoginException {
    logout();
    return true;
  }
  public boolean commit() throws javax.security.auth.login.LoginException {
    if (loggedIn) {
      if (!subject.getPrincipals().contains(principal))
        subject.getPrincipals().add(principal);
    }
    return true;
  }
  public boolean logout() throws javax.security.auth.login.LoginException {
    subject.getPrincipals().remove(principal);
    principal = null;
    loggedIn = false;
    return true;
  }
}
```

Obrázek 105. MyLogin.java

Obrázek 106 na stránce 485 je vzorový kód JAASLoginPrincipal.java, zkopírovaný do balíku security.jaas. Účelem příkazu JAASLoginPrincipal je implementovat rozhraní produktu

`java.security.Principal` za účelem uchování záznamů o uživatelích, kteří byli úspěšně přihlášeni pomocí produktu `MyLogin`.

```
package security.jaas;
public class JAASPrincipal implements java.security.Principal,
    java.io.Serializable {
  private static final long serialVersionUID = 1L;
  String                    name;
  public JAASPrincipal(String name) {
    this.name = name;
  }
  public String getName() {
    return name;
  }
  public String toString() {
    return (name);
  }
  public boolean equals(Object object) {
    if (object != null && object instanceof JAASPrincipal
        && name.equals(((JAASPrincipal) object).getName()))
      return true;
    else
      return false;
  }
  public int hashCode() {
    return name.hashCode();
  }
}
```

Obrázek 106. JAASLoginPrincipal.java

Kód v souboru PubSync.java , který je upraven tak, aby přidal `jméno uživatele` a `heslo` , je v produktu Obrázek 107 na stránce 485kurzívou.

```
package com.ibm.mq.id;
import com.ibm.micro.client.mqttv3.*;
public class PubSyncSSL {
  public static void main(String[] args) {
    try {
      MqttClient client = new MqttClient(Example.SSLAddress, Example.clientId);
      MqttTopic topic = client.getTopic(Example.topicString);
      MqttMessage message = new MqttMessage(Example.publication.getBytes());
      message.setQos(Example.QoS);
      MqttConnectOptions conOptions = new MqttConnectOptions();
      conOptions.setUserName(Example.username);
      conOptions.setPassword(Example.password);
      client.connect(conOptions);
      System.out.println("Waiting for up to " + Example.sleepTimeout / 1000
              + " seconds for publication of \"" + message.toString()
              + "\" with QoS = " + message.getQos());
      System.out.println("On topic \"" + topic.getName() + "\" for client instance: \""
              + client.getClientId() + "\" on address " + client.getServerURI() + "\"");
      System.out.println("With username \"" + conOptions.getUserName()
              + "\" and password \"" + String.valueOf(conOptions.getPassword()) + "\"");
      MqttDeliveryToken token = topic.publish(message);
      token.waitForCompletion(Example.sleepTimeout);
      System.out.println("Delivery token \"" + token.hashCode()
          + "\" has been received: " + token.isComplete());
      client.disconnect();
    } catch (Exception e) {
      System.out.println("Client exception caught");
      e.printStackTrace();
    }
  }
}
```

Obrázek 107. PubSyncJAAS.Java

Upravte konstanty v souboru Example.java tak, aby odpovídaly vaší konfiguraci. Ignoruje se nastavení SSL pro tento příklad.

```
package com.ibm.mq.id;
import java.util.Properties;
import java.util.UUID;
public final class Example {
  public static final String          TCPAddress =
    System.getProperty("TCPAddress", "tcp://localhost:1883");
  public static final String          SSLAddress =
    System.getProperty("SSLAddress", "ssl://localhost:8883");
  public static final String            username =
    System.getProperty("username", System.getProperty("user.name"));
  public static final char []           password =
    System.getProperty("password", "Password").toCharArray();
  public static String                  clientId =
  String.format("%-23.23s", username  + "_" +
    System.getProperty("clientId",
    (UUID.randomUUID().toString())).trim()).replace('-', '_');
  public static final String         topicString =
    System.getProperty("topicString", "MQTT Example");
  public static final String         publication =
    System.getProperty("publication", "Hello World " +
  String.format("%tc", System.currentTimeMillis()));
  public static final int         quiesceTimeout =
    Integer.parseInt(System.getProperty("timeout", "10000"));
  public static final int           sleepTimeout =
    Integer.parseInt(System.getProperty("timeout", "10000"));
  public static final boolean       cleanSession =
    Boolean.parseBoolean(System.getProperty("cleanSession", "false"));
  public static final int                   QoS =
    Integer.parseInt(System.getProperty("QoS", "1"));
  public static final boolean           retained =
    Boolean.parseBoolean(System.getProperty("retained", "false"));
  public static final Properties getSSLSettings() {
    final Properties properties = new Properties();
    properties.setProperty("com.ibm.ssl.keyStore",
        "C:\\IBM\\MQ\\Data\\ClientKeyStore.jks");
    properties.setProperty("com.ibm.ssl.keyStoreType",
        "JKS");
    properties.setProperty("com.ibm.ssl.keyStorePassword",
        "password");
    properties.setProperty("com.ibm.ssl.trustStore",
        "C:\\IBM\\MQ\\Data\\ClientTrustStore.jks");
    properties.setProperty("com.ibm.ssl.trustStoreType",
        "JKS");
    properties.setProperty("com.ibm.ssl.trustStorePassword",
        "password");
    return properties;
  }
}
```

Obrázek 108. Example.java

Ověřování připojení telemetrie SSL pomocí certifikátů podepsaných svým držitelem

Použijte certifikáty podepsané sebou samým pomocí produktu **Keytool** k ověření připojení přes SSL. Máte možnost ověřit kanál telemetrie nebo kanál telemetrie a klienty, které se k němu připojují. Zprávy tekoucí na připojení jsou šifrovány.

Než začnete

Do the task, “Vytvoření první aplikace MQ Telemetry Transport Publisher pomocí jazyka Java” na stránce 459 before you start, to get PubSync.java working with an unsecured TCP/IP connection. V této úloze upravíte PubSync.java pro práci s připojením SSL.

Informace o této úloze

Kroky v úloze se zapisují jako programovací cvičení. Chcete-li provést skutečnou autentizaci v produkčním prostředí, musíte upravit proceduru.

Úloha je zapsána pro systém Windows. Změňte cesty k adresářům pro produkt Linux.

Postup

1. Proveďte úlohu “Úprava souboru PubSync.java pro použití zabezpečení SSL” na stránce 487, chcete-li upravit soubor PubSync.java pro použití zabezpečení SSL.
2. Konfigurujte kanál telemetrie a vytvořte úložiště klíčů pro použití zabezpečení SSL.

   Ověřit pouze kanál telemetrie nebo kanál a klienty, kteří se k němu připojují:

   - Chcete-li se připojit k protokolu SSL a ověřit kanál telemetrie, proveďte úlohu “Ověřování kanálu telemetrie” na stránce 488.
   - Chcete-li se připojit k protokolu SSL a ověřit kanál telemetrie a klienty, kteří se k němu připojují, proveďte úlohu “Ověřování kanálu telemetrie a klientů” na stránce 489.
3. Zastavte a znovu spusťte službu telemetrie (MQXR), abyste vybrali změny v konfiguracích kanálu telemetrie.
4. Spusťte klientský program, abyste viděli, zda konfigurace funguje.

Úprava souboru PubSync.java pro použití zabezpečení SSL

Upravte první příklad programu vydavatele, chcete-li se připojit k telemetrickým kanálu pomocí zabezpečení SSL. Nastavte vlastnosti SSL používané upraveným programem.

Než začnete

Předpokládá se, že jste nainstalovali soubory JAR klienta protokolu MQTT v3 , dokumentaci Javadoc, Eclipse, nakonfigurované kanály telemetrie a před provedením této úlohy zakódované a spuštěné PubSync.java . Máte pracovní prostor Eclipse , který obsahuje spuštěnou verzi souboru PubSync.java.

Informace o této úloze

Úloha používá klienta vydavatele PubSync.java, který jste vytvořili v produktu “Vytvoření první aplikace MQ Telemetry Transport Publisher pomocí jazyka Java” na stránce 459 jako základ. Pouze malé úpravy jsou nezbytné pro použití SSL; viz Obrázek 109 na stránce 488 a Obrázek 110 na stránce 488.

Postup

1. Vytvořte kopii produktu PubSync.java v balíku com.ibm.mq.id a pojmenujte kopii PubSyncSSL.java.

   Postup vytvoření souboru PubSync.java v balíku produktu com.ibm.mq.id naleznete v tématu “Vytvoření první aplikace MQ Telemetry Transport Publisher pomocí jazyka Java” na stránce 459 .
2. Nastavte volbu Example.SSLAddress na adresu soketu kanálu telemetrie, který jste konfigurovali pro použití pro konfiguraci SSL.
3. Změňte parametr adresy soketu konstruktoru klienta na použití Example.SSLAddress.

```
MqttClient client = new MqttClient(Example.SSLAddress, Example.clientId);
```

4. Nastavte MqttConnectOptions.SSLProperties v PubSyncSSL.javaa předejte MqttConnectOptions jako parametr MqttClient.connect.

```
MqttConnectOptions conOptions = new MqttConnectOptions();
conOptions.setSSLProperties(Example.getSSLSettings());
client.connect(conOptions);
```

   Viz italizovaný kód PubSyncSSL.Java pomocí konstant nastavených v Example.java.

Příklady

Úpravy souboru PubSync.java pro přidání SSL jsou zobrazeny v souboru Obrázek 109 na stránce 488 kurzívou.

```
package com.ibm.mq.id;
import com.ibm.micro.client.mqttv3.*;
public class PubSyncSSL {
  public static void main(String[] args) {
    try {
      MqttClient client = new MqttClient(Example.SSLAddress, Example.clientId);
      MqttTopic topic = client.getTopic(Example.topicString);
      MqttMessage message = new MqttMessage(Example.publication.getBytes());
      message.setQos(Example.QoS);
      MqttConnectOptions conOptions = new MqttConnectOptions();
      conOptions.setSSLProperties(Example.getSSLSettings());
      client.connect(conOptions);
      System.out.println("Waiting for up to " + Example.sleepTimeout / 1000
              + " seconds for publication of \"" + message.toString()
              + "\" with QoS = " + message.getQos());
      System.out.println("On topic \"" + topic.getName() + "\" for client instance: \""
              + client.getClientId() + "\" on address " + client.getServerURI() + "\"");
      System.out.println("SSL Properties" + conOptions.getSSLProperties());
      MqttDeliveryToken token = topic.publish(message);
      token.waitForCompletion(Example.sleepTimeout);
      System.out.println("Delivery token \"" + token.hashCode()
          + "\" has been received: " + token.isComplete());
      client.disconnect();
    } catch (Exception e) {
      System.out.println("Client exception caught");
      e.printStackTrace();
    }
  }
}
```

Obrázek 109. PubSyncSSL.Java

Úpravy souboru Example.java se zobrazují v produktu Obrázek 110 na stránce 488.

```
 public static final String          SSLAddress =
    System.getProperty("SSLAddress", "ssl://localhost:8883");

 public static final Properties getSSLSettings() {
    final Properties properties = new Properties();
    properties.setProperty("com.ibm.ssl.keyStore", "C:\\Certificates\\SSClientKey.jks");
    properties.setProperty("com.ibm.ssl.keyStoreType", "JKS");
    properties.setProperty("com.ibm.ssl.keyStorePassword", "password");
    properties.setProperty("com.ibm.ssl.trustStore", "C:\\Certificates\\SSClientTrust.jks");
    properties.setProperty("com.ibm.ssl.trustStoreType", "JKS");
    properties.setProperty("com.ibm.ssl.trustStorePassword", "password");
    return properties;
```

Obrázek 110. Úpravy produktu Example.java

Ověřování kanálu telemetrie

Klienti ověřují kanál telemetrie, aby šifrovali obsah zpráv proudících na kanálu, a aby se ujistil, že se klient připojuje ke správnému kanálu telemetrie. Server neověřuje klienta.

Informace o této úloze

Pro vytváření a správu certifikátů podepsaných sebou samým můžete použít řadu různých editorů úložiště klíčů. Úloha používá příkaz s příkazovým řádkem **keytool** , který je součástí prostředí JRE. Nástroj grafického uživatelského rozhraní **iKeyman**, který je dodáván s produktem WebSphere MQ , můžete použít k procházení úložiště klíčů a generování klíčů. Spusťte příkaz **iKeyman** pomocí příkazu **strmqikm**.

Postup

1. Vytvořte kanál telemetrie SSLSSOptClients , který vyžaduje připojení SSL pomocí průvodce **Nový kanál telemetrie** . Kanál přijímá anonymní klienty.

   Přizpůsobte konfiguraci kanálu z následující sekce konfigurace. Neupravujte soubor vlastností telemetrie přímo; použijte průvodce.

   ```
   com.ibm.mq.MQXR.channel/SSLSSOptClients: \
   com.ibm.mq.MQXR.Port=8883;\
   com.ibm.mq.MQXR.Backlog=4096;\
   com.ibm.mq.MQXR.KeyFileName=C:\\Certificates\\SSServerOptKey.jks;\
   com.ibm.mq.MQXR.PassPhrase=password;\
   com.ibm.mq.MQXR.ClientAuth=OPTIONAL;\
   com.ibm.mq.MQXR.UserName=Admin;\
   com.ibm.mq.MQXR.StartWithMQXRService=true
   ```

2. Vygenerujte klíče pro klienta k ověření kanálu telemetrie.

   a) Vygenerujte dvojici klíčů s vlastním podpisem pro kanál telemetrie v novém úložišti klíčů SSServerOptKey.jks:

   ```
   Keytool -genkey -noprompt -alias SSServerPrivate
           -dname "CN=mqttserver.ibm.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
           -keystore SSServerOptKey.jks -storepass password -keypass password
   ```

   b) Vyexportujte jeho veřejný certifikát jako soubor ASCII pomocí volby -rfc :

   ```
   Keytool -export -noprompt -alias SSServerPrivate -file SSServerPublic.cer
           -keystore SSServerOptKey.jks -storepass password -rfc
   ```

   Pokud spouštíte úlohu v systému Windows, poklepejte na položku SSServerPublic.cer a zkontrolujte její obsah.

   c) Naimportujte veřejný certifikát do nového úložiště údajů o důvěryhodnosti klienta, SSClientTrust.jks:

   ```
   Keytool -import -noprompt -alias SSServerPublic -file SSServerPublic.cer
           -keystore SSClientTrust.jks -storepass password
   ```

   d) Vytvořte prázdné úložiště klíčů klienta, SSClientKey.jks.

   Keytool nemá příkaz k vytvoření prázdného úložiště klíčů. Můžete vybrat ze dvou voleb:

   i) Spusťte produkt **strmqikm**a vytvořte úložiště klíčů SSClientKey.jks, ale nepřidávejte žádné klíče.

   ii) Proveďte krok 3a v produktu “Ověřování kanálu telemetrie a klientů” na stránce 489, ale ještě nepoužívejte klíče.

Ověřování kanálu telemetrie a klientů

Klienti ověřují kanál telemetrie a kanál telemetrie ověří připojení klientů k němu. Zprávy přicházející do kanálu jsou šifrovány.

Informace o této úloze

Pro vytváření a správu certifikátů podepsaných sebou samým můžete použít řadu různých editorů úložiště klíčů. Úloha používá příkaz s příkazovým řádkem **keytool** , který je součástí prostředí JRE. Nástroj grafického uživatelského rozhraní **iKeyman**, který je dodáván s produktem WebSphere MQ , můžete použít k procházení úložiště klíčů a generování klíčů. Spusťte příkaz **iKeyman** pomocí příkazu **strmqikm**.

Kanál telemetrie je konfigurován s použitím jiného úložiště klíčů do úlohy “Ověřování kanálu telemetrie” na stránce 488. Můžete použít stejné úložiště klíčů a vynechat krok “2” na stránce 490 , chcete-li přidat klíče do úložiště klíčů.

Postup

1. Vytvořte kanál telemetrie SSLSSReqClients , který vyžaduje připojení SSL pomocí průvodce **Nový kanál telemetrie** . Kanál přijímá pouze ověřené klienty.

   Přizpůsobte konfiguraci kanálu z následující stanzy konfigurace:

   ```
   com.ibm.mq.MQXR.channel/SSLSSReqClients: \
   com.ibm.mq.MQXR.Port=8884;\
   com.ibm.mq.MQXR.Backlog=4096;\
   com.ibm.mq.MQXR.KeyFileName=C:\\Certificates\\SSServerReqKey.jks;\
   com.ibm.mq.MQXR.PassPhrase=password;\
   com.ibm.mq.MQXR.ClientAuth=REQUIRED;\
   com.ibm.mq.MQXR.UserName=Admin;\
   com.ibm.mq.MQXR.StartWithMQXRService=true
   ```

2. Vygenerujte klíče pro klienta k ověření kanálu telemetrie.

   a) Vygenerujte dvojici klíčů s vlastním podpisem pro kanál telemetrie v novém úložišti klíčů SSServerReqKey.jks:

   ```
   Keytool -genkey -noprompt -alias SSServerPrivate
           -dname "CN=mqttserver.ibm.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
           -keystore SSServerReqKey.jks -storepass password -keypass password
   ```

   b) Vyexportujte jeho veřejný certifikát jako soubor ASCII pomocí volby -rfc :

   ```
   Keytool -export -noprompt -alias SSServerPrivate -file SSServerPublic.cer
           -keystore SSServerReqKey.jks -storepass password -rfc
   ```

   Pokud spouštíte úlohu v systému Windows, poklepejte na položku SSServerPublic.cer a zkontrolujte její obsah.

   c) Naimportujte veřejný certifikát do nového úložiště údajů o důvěryhodnosti klienta SSClientTrust.jks:

   ```
   Keytool -import -noprompt -alias SSServerPublic -file SSServerPublic.cer
           -keystore SSClientTrust.jks -storepass password
   ```

3. Vygenerujte klíče pro kanál telemetrie k ověření klienta.

   a) Vygenerujte dvojici klíčů podepsaný (svým) držitelem pro klienta v novém úložišti klíčů, SSClientKey.jks:

   ```
   Keytool -genkey -noprompt -alias SSClientPrivate
           -dname "CN=mqttclient.ibm.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
           -keystore SSClientKey.jks -storepass password -keypass password
   ```

   b) Vyexportujte jeho veřejný certifikát jako soubor ASCII pomocí volby -rfc :

   ```
   Keytool -export -noprompt -alias SSClientPrivate -file SSClientPublic.cer
           -keystore SSClientKey.jks -storepass password -rfc
   ```

   Pokud spouštíte úlohu v systému Windows, poklepejte na položku SSClientPublic.cer a zkontrolujte její obsah.

   c) Importujte veřejný certifikát do úložiště klíčů serveru SSServerReqKey.jks:

   ```
   Keytool -import -noprompt -alias SSClientPublic -file SSClientPublic.cer
           -keystore SSServerReqKey.jks -storepass password
   ```

   Kanály telemetrie používají stejné úložiště pro soukromé klíče i důvěryhodné certifikáty.

Ověřování připojení telemetrie SSL pomocí řetězce certifikátů

Použijte podepsané certifikáty získané od certifikační autority nebo z implementace svého vlastního certifikačního postupu k ověření připojení přes SSL. Máte možnost ověřit kanál telemetrie nebo kanál telemetrie a klienty, které se k němu připojují. Zprávy tekoucí na připojení jsou šifrovány.

Než začnete

Do the task, "Ověřování připojení telemetrie SSL pomocí certifikátů podepsaných svým držitelem" na stránce 486 before you start, to get PubSyncSSL.Java working with a secured TCP/IP connection using self-signed certificates.

Informace o této úloze

V této úloze můžete upravit úlohy "Ověřování kanálu telemetrie" na stránce 488a "Ověřování kanálu telemetrie a klientů" na stránce 489 v produktu "Ověřování připojení telemetrie SSL pomocí certifikátů podepsaných svým držitelem" na stránce 486pro práci s klíči certifikovaným řetězcem certifikátů.

Můžete buď získat certifikáty pro tuto úlohu od certifikační autority, nebo můžete pomocí webových serverů, jako je http://www.openca.org/ , získat certifikáty. Komerční certifikační autority obecně poskytují zkušební certifikáty na krátkou dobu bez poplatku. Tato úloha byla testována pomocí komerčně získaných certifikátů.

Další možností je vytvořit vlastní certifikační proces a spustit jej na vašich vlastních počítačích pomocí nástrojů z webů, jako je https://www.openssl.org/.

Důvěryhodné úložiště prostředí JRE `cacerts` se v této úloze nepoužívají. Úložiště údajů o důvěryhodnosti prostředí JRE `cacerts` můžete použít na klientu v úloze "Ověřování kanálu telemetrie" na stránce 491místo použití uvedeného úložiště údajů o důvěryhodnosti. Řetěz certifikátů může být podepsán dobře známou certifikační autoritou, která již má svůj kořenový certifikát v úložišti `cacerts` na straně klienta. V takovém případě neuvádějte úložiště údajů o důvěryhodnosti na klientovi. Ujistěte se, že v klientu je nainstalováno více prostředí JRE, které spravujete ve správném úložišti `cacerts` .

Postup

1. Pokud jste tak dosud neučinili, proveďte úlohu "Úprava souboru PubSync.java pro použití zabezpečení SSL" na stránce 487, abyste upravili soubor PubSync.java pro použití zabezpečení SSL.
2. Konfigurujte kanál telemetrie a vytvořte úložiště klíčů pro použití zabezpečení SSL.

   Ověřit pouze kanál telemetrie nebo kanál a klienty, kteří se k němu připojují:

   - Chcete-li se připojit k protokolu SSL a ověřit kanál telemetrie, proveďte úlohu "Ověřování kanálu telemetrie" na stránce 491.
   - Chcete-li se připojit k protokolu SSL a ověřit kanál telemetrie a klienty, kteří se k němu připojují, proveďte úlohu "Ověřování kanálu telemetrie a klientů" na stránce 493.
3. Zastavte a znovu spusťte službu telemetrie (MQXR), abyste vybrali změny v konfiguracích kanálu telemetrie.
4. Spusťte klientský program, abyste viděli, zda konfigurace funguje.

Ověřování kanálu telemetrie

Klienti ověřují kanál telemetrie, aby šifrovali obsah zpráv proudících na kanálu, a aby se zajistilo, že se klient připojí ke správnému kanálu telemetrie. Server neověřuje klienta.

Informace o této úloze

Pro vytváření a správu certifikátů můžete použít řadu různých editorů úložiště klíčů. Úloha používá příkaz s příkazovým řádkem **`keytool`** , který je součástí prostředí JRE. Nástroj grafického uživatelského rozhraní **`iKeyman`**, který je dodáván s produktem WebSphere MQ , můžete použít k procházení úložiště klíčů a generování klíčů. Spusťte příkaz **`iKeyman`** pomocí příkazu **`strmqikm`**.

Postup

1. Vytvořte kanál telemetrie `SSLCAOptClients` , který vyžaduje připojení SSL pomocí průvodce **Nový kanál telemetrie** . Kanál přijímá anonymní klienty.

Přizpůsobte konfiguraci kanálu z následující sekce konfigurace. Neupravujte soubor vlastností telemetrie přímo; použijte průvodce.

```
com.ibm.mq.MQXR.channel/SSLCAOptClients: \
com.ibm.mq.MQXR.Port=8885;\
com.ibm.mq.MQXR.Backlog=4096;\
com.ibm.mq.MQXR.KeyFileName=C:\\Certificates\\CAServerOptKey.jks;\
com.ibm.mq.MQXR.PassPhrase=password;\
com.ibm.mq.MQXR.ClientAuth=OPTIONAL;\
com.ibm.mq.MQXR.UserName=Admin;\
com.ibm.mq.MQXR.StartWithMQXRService=true
```

2. Vygenerujte klíč podepsaný certifikační autoritou pro klienta pro ověření kanálu telemetrie.

   a) Vygenerujte dvojici klíčů s vlastním podpisem pro kanál telemetrie v novém úložišti klíčů `SSServerOptKey.jks`:

   ```
   Keytool -genkey -noprompt -alias CAServerPrivate -keyalg RSA
           -dname "CN=mqttserverOpt.ibm.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
           -keystore CAServerOptKey.jks -storepass password -keypass password
   ```

   Algoritmus klíče je nastaven na RSA, protože jej některé certifikační autority vyžadují. Obecný název certifikátu musí být jedinečný, některé certifikační autority nevydávají klíče s identickými společnými názvy.

   b) Vytvoření žádosti o podpis certifikátu (CSR) jako souboru ASCII

   ```
   Keytool -certreq -noprompt -alias CAServer -file CAServerOptKey.csr
           -dname "CN=mqttserverReq.ibm.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
           -keystore CAServerOptKey.jks -storepass password -keypass password
   ```

   c) Spusťte software certifikační autority nebo přihlášení na jejich web. Vložte do obsahu souboru `CAServerOptKey.csr` , je-li dotázán na soubor CSR.

   d) Certifikační autorita vrací jeden nebo dva certifikáty a podepsaný soubor odpovědí jako soubory ASCII. Vložte obsah do dvou nebo tří souborů:

   kořenový certifikát
   Vložit do `CARoot.cer`

   intermediační certifikát
   Vložit do `CAInter.cer`

   Podepsaný soubor odpovědí serveru
   Vložit do `CAServerOpt.rsp`

   Úložiště certifikátů prostředí JRE se v této úloze nepoužívá. Pokud jste obdrželi jeden kořenový certifikát a podepsanou odpověď od CA, použijte kořenový certifikát a podepsanou odpověď v následujících krocích. Pokud jste obdrželi kořenový a přechodný certifikát, použijte intermediační certifikát a podepsanou odpověď.

   e) Přijmout podepsanou odpověď serveru do úložiště klíčů serveru, ze kterého jste vydali žádost o certifikát.

   Přijímání odpovědi upraví certifikát podepsaný svým držitelem, aby byl podepsán CA. Podíváte-li se na certifikát v úložišti klíčů před a po přijetí odpovědi, změny podepisujícího subjektu se změní. Pokud tomu tak není, nástroj pro správu klíčů ohlásí chybu. Před použitím certifikátu jej zkontrolujte a ověřte, zda je podepisující subjekt nyní certifikační autoritou.

   ```
   Keytool -import -noprompt -alias CAServer -file CAServerOpt.rsp
           -keystore CAServerOptKey.jks -storepass password
   ```

   V nějakém softwaru pro správu klíčů, jako je například **iKeyman**, se soubory odpovědí neimportují, nýbrž spíše než importujete.

   f) Importujte certifikát CA do úložiště údajů o důvěryhodnosti klienta.

   Importujte buď přechodný certifikát, pokud jste od CA obdrželi dvě certifikáty, nebo pokud jste obdrželi pouze jeden certifikát, nebo kořenový certifikát.

Proveďte jednu z následujících akcí:

```
keytool -import -alias CAInter -file CAInter.cer
        -keystore CAClientTrust.jks -storepass password
```

Nebo:

```
keytool -import -alias CARoot -file CARoot.cer
        -keystore CAClientTrust.jks -storepass password
```

Ověřování kanálu telemetrie a klientů

Klienti ověřují kanál telemetrie a kanál telemetrie ověří připojení klientů k němu. Zprávy přicházející do kanálu jsou šifrovány.

Informace o této úloze

Pro vytváření a správu certifikátů můžete použít řadu různých editorů úložiště klíčů. Úloha používá příkaz s příkazovým řádkem **keytool** , který je součástí prostředí JRE. Nástroj grafického uživatelského rozhraní **iKeyman**, který je dodáván s produktem WebSphere MQ , můžete použít k procházení úložiště klíčů a generování klíčů. Spusťte příkaz **iKeyman** pomocí příkazu **strmqikm**.

Kanál telemetrie je nakonfigurován s jiným úložištěm klíčů pro jednu v úloze, “Ověřování kanálu telemetrie” na stránce 491. Můžete použít stejné úložiště klíčů a vynechat krok “2” na stránce 493 , chcete-li přidat klíče do úložiště klíčů.

Postup

1. Vytvořte kanál telemetrie SSLCAReqClients , který vyžaduje připojení SSL pomocí průvodce **Nový kanál telemetrie** . Kanál přijímá pouze ověřené klienty.

   Přizpůsobte konfiguraci kanálu z následující sekce konfigurace. Neupravujte soubor vlastností telemetrie přímo; použijte průvodce.

   ```
   com.ibm.mq.MQXR.channel/SSLCAReqClients: \
   com.ibm.mq.MQXR.Port=8886;\
   com.ibm.mq.MQXR.Backlog=4096;\
   com.ibm.mq.MQXR.KeyFileName=C:\\Certificates\\CAServerReqKey.jks;\
   com.ibm.mq.MQXR.PassPhrase=password;\
   com.ibm.mq.MQXR.ClientAuth=REQUIRED;\
   com.ibm.mq.MQXR.UserName=Admin;\
   com.ibm.mq.MQXR.StartWithMQXRService=true
   ```

2. Vygenerujte klíč podepsaný certifikační autoritou pro klienta pro ověření kanálu telemetrie.

   a) Vygenerujte dvojici klíčů s vlastním podpisem pro kanál telemetrie v novém úložišti klíčů CAServerReqKey.jks:

   ```
   Keytool -genkey -noprompt -alias CAServerPrivate -keyalg RSA
           -dname "CN=mqttserverReq.ibm.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
           -keystore CAServerReqKey.jks -storepass password -keypass password
   ```

   Algoritmus klíče je nastaven na RSA, protože jej některé certifikační autority vyžadují. Obecný název certifikátu musí být jedinečný, některé certifikační autority nevydávají klíče s identickými společnými názvy.

   b) Vytvoření žádosti o podpis certifikátu (CSR) jako souboru ASCII

   ```
   Keytool -certreq -noprompt -alias CAServer -file CAServerReqKey.csr
           -dname "CN=mqttserverReq.ibm.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
           -keystore CAServerReqKey.jks -storepass password -keypass password
   ```

   c) Spusťte software certifikační autority nebo přihlášení na jejich web. Vložte do obsahu souboru CAServerReqKey.csr , je-li dotázán na soubor CSR.

d) Certifikační autorita vrací jeden nebo dva certifikáty a podepsaný soubor odpovědí jako soubory ASCII. Vložte obsah do dvou nebo tří souborů:

kořenový certifikát
Vložit do `CARoot.cer`

intermediační certifikát
Vložit do `CAInter.cer`

Podepsaný soubor odpovědí serveru
Vložit do `CAServerReq.rsp`

Úložiště certifikátů prostředí JRE se v této úloze nepoužívá. Pokud jste obdrželi jeden kořenový certifikát a podepsanou odpověď od CA, použijte kořenový certifikát a podepsanou odpověď v následujících krocích. Pokud jste obdrželi kořenový a přechodný certifikát, použijte intermediační certifikát a podepsanou odpověď.

e) Přijmout podepsanou odpověď serveru do úložiště klíčů serveru, ze kterého jste vydali žádost o certifikát.

Přijímání odpovědi upraví certifikát podepsaný svým držitelem, aby byl podepsán CA. Podíváte-li se na certifikát v úložišti klíčů před a po přijetí odpovědi, změny podepisujícího subjektu se změní. Pokud tomu tak není, nástroj pro správu klíčů ohlásí chybu. Před použitím certifikátu jej zkontrolujte a ověřte, zda je podepisující subjekt nyní certifikační autoritou.

```
Keytool -import -noprompt -alias CAServer -file CAServerReq.rsp
        -keystore CAServerReqKey.jks -storepass password
```

V nějakém softwaru pro správu klíčů, jako je například **iKeyman**, se soubory odpovědí neimportují, nýbrž spíše než importujete.

f) Importujte certifikát CA do úložiště údajů o důvěryhodnosti klienta.

Importujte buď přechodný certifikát, pokud jste od CA obdrželi dvě certifikáty, nebo pokud jste obdrželi pouze jeden certifikát, nebo kořenový certifikát.

Proveďte jednu z následujících akcí:

```
keytool -import -alias CAInter -file CAInter.cer
        -keystore CAClientTrust.jks -storepass password
```

Nebo:

```
keytool -import -alias CARoot -file CARoot.cer
        -keystore CAClientTrust.jks -storepass password
```

3. Vygenerujte klíč CA pro kanál telemetrie k ověření klientů.

a) Vygenerujte dvojici klíčů podepsaný (svým) držitelem pro klienty v novém úložišti klíčů `CAClientKey.jks`:

```
Keytool -genkey -noprompt -alias CAClientPrivate -keyalg RSA
        -dname "CN=mqttserverReq.ibm.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
        -keystore CAClientKey.jks -storepass password -keypass password
```

Algoritmus klíče je nastaven na RSA, protože jej některé certifikační autority vyžadují. Obecný název certifikátu musí být jedinečný, některé certifikační autority nevydávají klíče s identickými společnými názvy.

b) Vytvoření žádosti o podpis certifikátu (CSR) jako souboru ASCII

```
Keytool -certreq -noprompt -alias CAClient -file CAClientKey.csr
        -dname "CN=mqttserverReq.ibm.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
        -keystore CAClientKey.jks -storepass password -keypass password
```

c) Spusťte software certifikační autority nebo přihlášení na jejich web. Vložte do obsahu souboru `CAClientKey.csr` , je-li dotázán na soubor CSR.

d) Certifikační autorita vrací jeden nebo dva certifikáty a podepsaný soubor odpovědí jako soubory ASCII. Vložte obsah do dvou nebo tří souborů:

kořenový certifikát
Vložit do `CARoot.cer`

intermediační certifikát
Vložit do `CAInter.cer`

Podepsaný soubor odpovědí klienta
Vložit do `CAClient.rsp`

Úložiště certifikátů prostředí JRE se v této úloze nepoužívá. Pokud jste obdrželi jeden kořenový certifikát a podepsanou odpověď od CA, použijte kořenový certifikát a podepsanou odpověď v následujících krocích. Pokud jste obdrželi kořenový a přechodný certifikát, použijte intermediační certifikát a podepsanou odpověď.

e) Přijmout podepsanou odpověď klienta do úložiště klíčů klienta, ze kterého jste vydali žádost o certifikát.

Přijímání odpovědi upraví certifikát podepsaný svým držitelem, aby byl podepsán CA. Podíváte-li se na certifikát v úložišti klíčů před a po přijetí odpovědi, změny podepisujícího subjektu se změní. Pokud tomu tak není, nástroj pro správu klíčů ohlásí chybu. Před použitím certifikátu jej zkontrolujte a ověřte, zda je podepisující subjekt nyní certifikační autoritou.

```
Keytool -import -noprompt -alias CAClient -file CAClient.rsp
        -keystore CAClientKey.jks -storepass password
```

V nějakém softwaru pro správu klíčů, jako je například **iKeyman**, se soubory odpovědí neimportují, nýbrž spíše než importujete.

f) Importujte certifikát CA do úložiště klíčů serveru.

Importujte buď přechodný certifikát, pokud jste od CA obdrželi dvě certifikáty, nebo pokud jste obdrželi pouze jeden certifikát, nebo kořenový certifikát.

Proveďte jednu z následujících akcí:

```
keytool -import -alias CAInter -file CAInter.cer
        -keystore CAServerReqKey.jks -storepass password
```

Nebo:

```
keytool -import -alias CARoot -file CARoot.cer
        -keystore CAServerReqKey.jks -storepass password
```

Vytvoření první aplikace MQ Telemetry Transport Publisher pomocí jazyka C

Postup vytvoření aplikace vydavatele klienta MQTT je popsán ve výukovém programu. Každý řádek kódu C je vysvětlen. Na konci úlohy jste vytvořili vydavatele MQTT.

Než začnete

Vyvinovaná klientská aplikace používá knihovny klienta MQTT v3 C klienta. Aplikace se připojuje k démonu WebSphere MQ Telemetry pro zařízení k publikování zpráv. Příklad komunikace klienta s produktem WebSphere MQ Telemetryviz téma Vytvoření prvního vydavatele .

Informace o této úloze

Příkladem je publikační aplikace, `pubsync.c`. Program `pubsync.c` publikuje zprávu s informačním obsahem `Hello World!` na téma `MQTT Examplea` čeká na potvrzení, že publikování bylo doručeno démonovi.

Pro zjednodušení nejsou návratové kódy z některých použitých funkcí testovány na správné dokončení. V produkčním kódu lze zkontrolovat návratové kódy, abyste se ujistili, že se program chová podle očekávání. Pokud dojde k neočekávané chybě, musí být provedena vhodná akce.

Nastavením odběratele na `MQTT Example` můžete zkontrolovat, zda aplikace funguje.

Použijte vybrané vývojové prostředí C pro vývoj, sestavení a spuštění klienta. Dáváte-li přednost, můžete kód kopírovat přímo z příkladů.

Postup

1. Vytvořte nový, prázdný zdrojový soubor, `pubsync.c`
2. Vytvořte soubor `settings.h`. Zkopírujte kód na obrázku 2 do souboru.

   Všechny parametry použité v programu jsou definovány v `settings.h`. Hodnoty můžete přepsat změnou hodnot v souboru.
3. Následující kroky popisují kód. Postupujte podle pokynů nebo zkopírujte kód z Obrázek 1 do `pubsync.c`.
4. Přidejte do souboru záhlaví příkazy include pro požadované standardní knihovny a soubory `MQTTClient.h` a `settings.h` .

   ```
   #include "stdio.h"
   #include "stdlib.h"
   #include "string.h"
   #include "MQTTClient.h"
   #include "settings.h"
   ```
5. Spusťte definici funkce `main()` .

   ```
   int main(int argc, char* argv[])
   {
   ```
6. Definujte lokální proměnné použité v programu.

   ```
   MQTTClient client;
   MQTTClient_connectOptions conn_opts = MQTTClient_connectOptions_initializer;
   MQTTClient_message pubmsg;
   MQTTClient_deliveryToken token;
   int rc;
   ```

   Poznámka: Volby připojení jsou vyžadovány funkcí `MQTTClient_connect` . `MQTTClient_connectOptions_initializer` obsahuje výchozí volby.
7. Vytvořte klienta.

   ```
   MQTTClient_create(&client, ADDRESS, CLIENTID, MQTTCLIENT_PERSISTENCE_NONE, NULL);
   ```

   - `& klient` je ukazatel na popisovač pro nově vytvořeného klienta. Když se tato funkce vrátí s návratovým kódem 0, obsahuje popisovač pro nového klienta. Příklad předpokládá úspěch. Otestujte kód chyby pro správné dokončení v produkčním kódu.
   - ADRESA je identifikátor URI portu MQTT, který démon monitoruje pro příchozí požadavky na připojení klienta.
   - `CLIENTID` je název používaný k identifikaci klienta pro démona. Každý aktivní klient musí mít jedinečný název. Pokud duplikujete identifikátor klienta ve dvou spuštěných klientech, dojde k výjimce v obou klientech a jeden z klientů se ukončí. Název je použit démonem k rozpoznání toho, že se připojuje k jthat klienta po odpojení, viz Identifikátor klienta.
   - Volba `MQTTCLIENT_PERSISTENCE_NONE` určuje, že se stav klienta nachází v paměti a je ztracen, dojde-li k selhání systému. `MQTTCLIENT_PERSISTENCE_DEFAULT` uvádí perzistenci založenou na systému souborů, poskytující určitou ochranu proti selháním. Pro více specializovaných aplikací můžete použít `MQTTCLIENT_PERSISTENCE_USER`, který poskytuje rozhraní pro implementaci vašeho vlastního mechanizmu perzistence. Další informace najdete v dokumentaci k rozhraním API pro produkt `MQTTClientPersistence.h`. Zda je perzistence vyžadována, je otázka návrhu aplikace. Další informace naleznete v tématu Perzistence zpráv.

- Výchozí démon TCP/IP démona pro MQTT je 1883. V tomto příkladu je výchozí adresa nastavena na `tcp://localhost:1883`.
- Dokud nezavoláte funkci `MQTTClient_connect` , neproběhne žádné zpracování zpráv.

8. Připojte klienta k démonu.

```
if ((rc = MQTTClient_connect(client, &conn_opts)) != MQTTCLIENT_SUCCESS) {
     printf("Failed to connect, return code %d\n", rc);
     exit(-1);
}
```

   - Je volána funkce `MQTTClient_connect` , předává popisovač klienta a ukazatel na volby připojení jako argumenty.
   - Návratový kód z volání `MQTTClient_connect` se testuje, aby se ujistil, že požadavek na připojení je úspěšný.
   - Pokud se `MQTTClient_connect` nezdaří, program skončí s kódem chyby -1.
   - Po připojení aplikace můžete začít publikovat a odebírat.
   - Malá zpráva "keep-alive" se odesílá každých 20 sekund, aby se zabránilo zavření připojení TCP/IP. Tato volba je nastavena příkazem `conn_opts.keepAliveInterval`.
   - Relace se spustí bez kontroly dokončení průběžných zpráv, které zbývají z předchozího připojení, protože `conn_opts.cleansession` je nastaven na hodnotu true. Další informace najdete v tématu Vyčistit relace.
   - Pro připojení se nevytvoří žádná poslední zpráva a zpráva o potvrzení. Další podrobnosti najdete v tématu Poslední podrobnosti a potvrzení.

9. Naplňte daty strukturu *MQTTClient_message* daty a definujte informační obsah zprávy a jeho atributy.

```
pubmsg.payload = PAYLOAD;
pubmsg.payloadlen = strlen(PAYLOAD);
pubmsg.qos = QOS;
pubmsg.retained = 0;
```

   - PAYLOAD je náš obsah zprávy.
   - Příklad používá informační obsah řetězce, ale informační obsah MQTT jsou bajtová pole. Délka řetězce je povinná pro uvedení velikosti informačního obsahu.
   - Příklad publikuje zprávu QoS=1 , takže nastavte hodnotu odpovídajícím způsobem.
   - Zachovaný atribut je nastaven na hodnotu false (0), protože zpráva nemá být uchována démonem. Další podrobnosti naleznete v tématu Zachovaná publikování.

10. Publikujte zprávu.

```
MQTTClient_publishMessage(client, TOPIC, &pubmsg, &token);
```

    - Funkce publikování určuje klienta, téma a informační obsah, který má být odeslán démonovi.
    - `TOPIC` je definováno v `settings.h` jako `MQTT Example`.
    - Funkci se také předává ukazatel na objekt *MQTTClient_deliveryToken*. Tento ukazatel se naplní tokenem představujícím zprávu, když se funkce vrátí.
    - Zpráva je nyní bezpečně přenesena do klienta MQTT, ale ještě nebyla přenesena do démona. Pokud má zpráva QoS=1 nebo 2, pak je zpráva uložena lokálně, v případě selhání klienta před dokončením doručení.
    - Tato funkce vrací kód chyby, který můžete otestovat pro správné dokončení ve výrobním kódu.

11. Čekejte na potvrzení ze serveru.

```
rc = MQTTClient_waitForCompletion(client, token, TIMEOUT);
```

    - Příklad příkazu `pubsync.c` čeká na potvrzení ze serveru, což potvrzuje, že zpráva byla doručena.
    - Argumenty klienta a tokenu označují specifickou zprávu, kterou program čeká na dokončení.

- Hodnota TIMEOUT omezuje dobu, po kterou program čeká na dokončení doručení zprávy. Úloha Vytvoření asynchronního vydavatele pro přenos MQ Telemetry pomocí jazyka C ukazuje, jak přijímat potvrzení bez čekání pomocí funkcí zpětného volání.
- Tato funkce vrací kód chyby, který může být testován pro správné dokončení v produkčním kódu.

12. Odpojte klienta od démona.

```
MQTTClient_disconnect(client, 10000);
```

- Klient se odpojí od serveru a čeká na všechny funkce zpětného volání (v tomto příkladu se nepoužívá), aby se dokončilo zpracování inflatézních zpráv.
- Druhý argument určuje časový limit uvedení do klidového stavu v milisekundách. Tento příklad čeká až 10 sekund na dokončení jakékoli jiné práce, kterou musí provést před odpojením.
- Tato funkce vrací kód chyby, který musí být testován pro správné dokončení v produkčním kódu.

13. Uvolnění paměti použité klientem a ukončení programu.

```
MQTTClient_destroy(&client);
}
```

Výsledky

Chcete-li zobrazit publikace odeslané tímto klientem, vytvořte odběratele tématu produktu MQTT Example . Další informace naleznete v tématu Vytvoření odběratele pro přenos MQ Telemetry pomocí jazyka C

Příklad

Obrázek 1 je úplný výpis kódu popsaného v tématu Procedura. Soubor settings.h na Obrázku 2 vám umožňuje změnit výchozí parametry použité v pubsync.c.

```
#include "stdio.h"
#include "stdlib.h"
#include "MQTTClient.h"
#include "settings.h"

int pubsync_main(int argc, char* argv[]) {
    MQTTClient client;
    MQTTClient_connectOptions conn_opts = MQTTClient_connectOptions_initializer;
    MQTTClient_message pubmsg;
    MQTTClient_deliveryToken token;
    int rc;

    MQTTClient_create(&client, ADDRESS, CLIENTID,  MQTTCLIENT_PERSISTENCE_NONE, NULL);
    if ((rc = MQTTClient_connect(client, &conn_opts)) != MQTTCLIENT_SUCCESS) {
          printf("Failed to connect, return code %d\n", rc);
         exit(-1);
    }
    pubmsg.payload = PAYLOAD;
    pubmsg.payloadlen = strlen(PAYLOAD);
    pubmsg.qos = QOS;
    pubmsg.retained = 0;
    MQTTClient_publishMessage(client, TOPIC, &pubmsg, &token);
    printf("Waiting for up to %d seconds for publication of %s\n"
          "on topic %s for client with ClientID: %s\n",
          TIMEOUT/1000, PAYLOAD, TOPIC, CLIENTID);
    rc = MQTTClient_waitForCompletion(client, token, TIMEOUT);
    printf("Message with delivery token %d delivered\n", token);
    MQTTClient_disconnect(client, 10000);
    MQTTClient_destroy(&client);
}
```

Obrázek 111. pubsync.c

```
#define ADDRESS   "tcp://localhost:1883"
#define CLIENTID  "ExampleClientPub"
#define TOPIC     "MQTT Example"
#define PAYLOAD   "Hello World!"
#define QOS      1
#define TIMEOUT   10000L
```

Obrázek 112. settings.h

Vytvoření asynchronního vydavatele pro přenos MQ Telemetry pomocí jazyka C

Postup pro vytvoření asynchronní aplikace vydavatele klienta MQTT je popsán ve výukovém programu. Každý řádek kódu C je vysvětlen. Na konci úlohy jste vytvořili asynchronní vydavatele MQTT.

V této úloze následujete výukový program k úpravě vaší první aplikace vydavatele. Úpravy umožňují aplikaci odesílat publikace bez čekání na potvrzení doručení. Potvrzení doručení jsou přijata funkcí zpětného volání, kterou vytvoříte.

Než začnete

Vyvinovaná klientská aplikace používá knihovny klienta MQTT v3 C klienta. Aplikace se připojuje k démonu WebSphere MQ Telemetry pro zařízení k publikování zpráv. Příklad komunikace klienta s produktem WebSphere MQ Telemetryviz téma Vytvoření prvního vydavatele .

Informace o této úloze

Příkladem je publikační aplikace, `pubasync.c`. Program `pubasync.c` publikuje zprávu s informačním obsahem `Hello World!` do tématu `MQTT Example`, aniž by čekal na potvrzení, že publikování bylo doručeno démonu. Potvrzení doručení se přijímají ve funkci zpětného volání, `MQTTClient_deliveryComplete`.

Pro zjednodušení nejsou návratové kódy z některých použitých funkcí testovány na správné dokončení. V produkčním kódu lze zkontrolovat návratové kódy, abyste se ujistili, že se program chová podle očekávání. Pokud dojde k neočekávané chybě, musí být provedena vhodná akce.

Nastavením odběratele na `MQTT Example` můžete zkontrolovat, zda aplikace funguje.

Použijte vybrané vývojové prostředí C pro vývoj, sestavení a spuštění klienta.

Kroky v Procedure upravují aplikaci `pubsync.c` z produktu “Vytvoření první aplikace MQ Telemetry Transport Publisher pomocí jazyka C” na stránce 495. Dáváte-li přednost, můžete kód kopírovat přímo z příkladů.

Postup

1. Vytvořte nový, prázdný zdrojový soubor, `callback.h`.
2. Zkopírujte kód na obrázku 2 do souboru.
   - `callback.h` deklaruje tři metody zpětného volání potřebné pro asynchronní operace klienta.
   - Proměnná, *deliveredtoken*, je také deklarována. K tomuto přístupu je přistupován hlavním programem a zpětným voláním na různých podprocesech provedení. Je proto prohlášen za nestálý. Když používáte zpětná volání, dbají na to, aby se k relevantním proměnným přistupovali při zajištění neporušenosti vláken.
3. Vytvořte nový, prázdný zdrojový soubor, `callback.c`.
4. Zkopírujte kód na obrázku 3 do souboru.
   - Produkt `callback.c` implementuje tři metody zpětného volání používané klientem pro asynchronní operace, `delivered`, `msgarrvda` `connlost`.
5. Přidejte příkaz include pro `callback.h` za ostatními zahrnuti do `pubasync.c`.

```
#include "callback.h"
```

6. Zkopírujte obsah souboru `pubsync.c` do nového souboru `pubasync.c`.
7. Těsně před voláním funkce `MQTTClient_connect` v `pubasync.c`nastavte metody zpětného volání pro klienta.

```
MQTTClient_setCallbacks(client, NULL, connlost, msgarrvd, delivered);
```

   - Je třeba určit tři funkce zpětného volání. Tyto funkce jsou implementovány v produktu `callback.c`.
   - `MQTTClient_messageArrived` se volá, když se odešle zpráva klientovi kvůli odpovídajícímu odběru. To musí vrátit hodnotu true, když byla přijatá zpráva úspěšně přijata klientskou aplikací. Navrácení nepravdivých informací informuje klienta o tom, že aplikace měla problém se zprávou.
   - `MQTTClient_connectionLost` se vyvolá, když klient ztratí své připojení k serveru.
   - Produkt `MQTTClient_deliveryComplete` se vyvolá, když server QoS1 nebo QoS2 dorazila a byla potvrzena serverem. Nevolá se pro zprávy QoS0 . V tomto příkladu tato funkce uloží token z doručené zprávy do *deliveredtoken* , aby indikoval, že byla doručena zpráva.
   - Server `MQTTClient_setCallbacks` musí být volán, když je klient odpojen od serveru.
   - Druhý argument umožňuje předat kontextové informace do funkcí zpětného volání. Tento parametr se v příkladu nepoužívá, takže je nastaven na hodnotu NULL.
8. Okamžitě před voláním na `MQTTClient_publishMessage`zrušte zaškrtnutí *deliveredtoken*. `MQTTClient_deliveryComplete` nastavuje *deliveredtoken* , když je přijat token.

```
deliveredtoken = 0;
```

9. Odeberte volání `MQTTClient_waitForCompletion` a následující příkaz `printf` a nahraďte smyčku s čekáním na shodu originálního tokenu a tokenu přijatého ve zpětném volání.

```
while(deliveredtoken != token);
```

   Toto je příklad a necope se s řadou situací, které musí být umístěny v návrhu provozního kódu. Tyto situace zahrnují:

   - V případě, že doručení není dokončeno, lze implementovat časový limit
   - Vícenásobné zprávy mohou být inflaténě. Ukázkový program povoluje v daném okamžiku kontrolovat pouze jeden token doručení.
10. Odpojte klienta od démona.

```
MQTTClient_disconnect(client, 10000);
```

    - Klient se odpojí od serveru a čeká na všechny funkce zpětného volání pro dokončení inflaténích zpráv.
    - Druhý argument určuje časový limit uvedení do klidového stavu v milisekundách. Tento příklad čeká až 10 sekund na dokončení jakékoli jiné práce, kterou musí provést před odpojením.
    - Tato funkce vrací kód chyby, který musí být testován pro správné dokončení v produkčním kódu.
11. Uvolnění paměti použité klientem a ukončení programu.

```
MQTTClient_destroy(&client);
}
```

Výsledky

Chcete-li zobrazit publikování odeslanou tímto klientem, vytvořte odběratele pro téma `MQTT Example` . Další podrobnosti viz téma Vytvoření odběratele pro produkt MQ Telemetry Transport .

Příklad

pubasync.c, callbacks.c a callbacks.h jsou úplné výpisy kódu popsaného v tématu Procedura.

```
#include "stdio.h"
#include "stdlib.h"
#include "MQTTClient.h"
#include "settings.h"
#include "callback.h"

int main(int argc, char* argv[]) {
    MQTTClient client;
    MQTTClient_connectOptions conn_opts = MQTTClient_connectOptions_initializer;
    MQTTClient_message pubmsg;
    MQTTClient_deliveryToken token;
    int rc;

    MQTTClient_create(&client, ADDRESS, CLIENTID, MQTTCLIENT_PERSISTENCE_NONE, NULL);
    MQTTClient_setCallbacks(client, NULL, connlost, msgarrvd, delivered);
    if ((rc = MQTTClient_connect(client, &conn_opts)) != MQTTCLIENT_SUCCESS) {
        printf("Failed to connect, return code %d\n", rc);
        exit(-1);
    }
    pubmsg.payload = PAYLOAD;
    pubmsg.payloadlen = strlen(PAYLOAD);
    pubmsg.qos = QOS;
    pubmsg.retained = 0;
    deliveredtoken = 0;
    MQTTClient_publishMessage(client, TOPIC, &pubmsg, &token);
    printf("Waiting for publication of %s\n"
           "on topic %s for client with ClientID: %s\n", PAYLOAD, TOPIC, CLIENTID);
    while(deliveredtoken != token);
    MQTTClient_disconnect(client, 10000);
    MQTTClient_destroy(&client);
}
```

Obrázek 113. pubasync.c

```
MQTTClient_deliveryComplete delivered;
MQTTClient_messageArrived msgarrvd;
MQTTClient_connectionLost connlost;

extern volatile MQTTClient_deliveryToken deliveredtoken;
```

Obrázek 114. callback.h

```
#include "MQTTClient.h"

volatile MQTTClient_deliveryToken deliveredtoken;

void delivered(void *context, MQTTClient_deliveryToken dt)
{
    printf("Message with token value %d delivery confirmed\n", dt);
    deliveredtoken = dt;
}

int msgarrvd(void *context, char *topicName, int topicLen, MQTTClient_message *message)
{
    int i;
    char* payloadptr;

    printf("Message arrived\n");
    printf("     topic: %s\n", topicName);
    printf("   message: ");

    payloadptr = message->payload;
    for(i=0; i<message->payloadlen; i++) {
        putchar(*payloadptr++);
    }
    putchar('\n');
    MQTTClient_freeMessage(&message);
    free(topicName);
    return 1;
}

void connlost(void *context, char *cause)
{
    printf("\nConnection lost\n");
    printf("     cause: %s\n", cause);
}
```

Obrázek 115. `callback.c`

```
#define ADDRESS    "tcp://localhost:1883"
#define CLIENTID   "ExampleClientPub"
#define TOPIC      "MQTT Example"
#define PAYLOAD    "Hello World!"
#define QOS        1
#define TIMEOUT    10000L
```

Obrázek 116. `settings.h`

Vytvoření odběratele pro přenos MQ Telemetry pomocí jazyka C

Kroky k vytvoření aplikace odběratele klienta MQTT jsou popsány ve výukovém programu. Každý řádek kódu C je vysvětlen. Na konci úlohy jste vytvořili odběratele MQTT.

Než začnete

Vyvinovaná klientská aplikace používá knihovny klienta MQTT v3 C klienta. Aplikace se připojuje k démonu WebSphere MQ Telemetry pro zařízení k publikování zpráv. Příklad komunikace klienta s produktem WebSphere MQ Telemetryviz téma Vytvoření prvního vydavatele .

Informace o této úloze

Příkladem je aplikace odběratele, `subscribe.c`. Program `subscribe.c` se přihlásí k odběru tématu `MQTT Example` a čeká na publikování, která odpovídají odběru, dokud uživatel neukončí program.

Odběratel vytvoří odběr na téma a čeká na zprávy, které odpovídají tématu odběru. Zprávy publikované během odpojení klienta, které odpovídají odběru vytvořenému klientem, mohou být přijaty při opětovném připojení klienta. Služba WebSphere MQ telemetrie (MQXR) nebo démon pro zařízení rozpoznává klienta, který byl již dříve připojen k identifikátoru klienta. Další informace naleznete v tématu Identifikátor klienta. Logický atribut `MQTTClient_connectOptions.cleansession` řídí, zda byly dříve odeslané publikování přijaty nebo ne. Další informace naleznete v tématu “Vyčistit relace” na stránce 511.

Pro zjednodušení nejsou návratové kódy z některých použitých funkcí testovány na správné dokončení. V produkčním kódu lze zkontrolovat návratové kódy, abyste se ujistili, že se program chová podle očekávání. Pokud se vyskytne neočekávaná chyba, lze provést odpovídající akci.

Dříve popsané ukázkové programy publikování můžete použít k odeslání odpovídajících publikování do démona produktu WebSphere MQ Telemetry pro zařízení. Můžete také použít průzkumníka produktu WebSphere MQ k vytvoření testovacích publikování na téma `MQTT Example` , pokud chcete připojit klienta k kanálu produktu WebSphere MQ Telemetry .

Pokyny v části Procedura předpokládají, že jste již vytvořili soubory `callback.c`, `callback.h` a `settings.h` v jedné z dřívějších úloh.

Použijte vybrané vývojové prostředí C pro vývoj, sestavení a spuštění klienta. Dáváte-li přednost, můžete kód kopírovat přímo z příkladů.

Postup

1. Vytvořte kopii produktu `settings.h` pro tento příklad a změňte hodnotu *CLIENTID* na následující příkaz:

   ```
   #define CLIENTID "ExampleClientSub"
   ```

   - Pokud se dva klienti se stejným ID pokusí připojit k jednomu serveru, jeden z nich je násilně odpojen. Typicky je nový pokus o připojení úspěšný a starší připojení je odpojeno.
   - Změna `ClientID` vám umožňuje použít dříve vytvořené příklady publikování k odeslání zpráv tomuto odběrateli.
2. Vytvořte nový, prázdný zdrojový soubor, `subscribe.c`.
3. Následující kroky popisují kód. Postupujte podle kroků nebo zkopírujte kód z produktu Obrázek 117 na stránce 506 do souboru `subscribe.c`.
4. Přidejte do souboru záhlaví příkazy include pro požadované standardní knihovny a soubory `MQTTClient.h` a `settings.h` .

   ```
   #include "stdio.h"
   #include "stdlib.h"
   #include "MQTTClient.h"
   #include "settings.h"
   ```
5. Spusťte definici funkce `main()` .

   ```
   int main(int argc, char* argv[]) {
   ```
6. Definujte lokální proměnné použité v programu.

   ```
   MQTTClient client;
   MQTTClient_connectOptions conn_opts = MQTTClient_connectOptions_initializer;
   MQTTClient_deliveryToken token;
   int rc;
   ```

   Volby připojení jsou vyžadovány funkcí `MQTTClient_connect` . `MQTTClient_connectOptions_initializer` obsahuje výchozí volby.
7. Vytvořte klienta.

   ```
   MQTTClient_create(&client, ADDRESS, CLIENTID, MQTTCLIENT_PERSISTENCE_NONE, NULL);
   ```

   - `& klient` je ukazatel na popisovač pro nově vytvořeného klienta. Když se tato funkce vrátí s návratovým kódem 0, ukazatel obsahuje popisovač nového klienta. Příklad předpokládá úspěch. Kód chyby může být testován pro správné dokončení v produkčním kódu.
   - ADRESA je identifikátor URI portu MQTT, který démon monitoruje pro příchozí požadavky na připojení klienta.
   - `CLIENTID` je název používaný k identifikaci klienta pro démona. Každý aktivní klient musí mít jedinečný název. Pokud duplikujete identifikátor klienta ve dvou spuštěných klientech, dojde

k výjimce v obou klientech a jeden z klientů se ukončí. Název je použit démonem k rozpoznání připojení klienta po odpojení, viz Identifikátor klienta.

- Volba `MQTTCLIENT_PERSISTENCE_NONE` určuje, že se stav klienta nachází v paměti a je ztracen, dojde-li k selhání systému. Volba `MQTTCLIENT_PERSISTENCE#_DEFAULT` určuje perzistenci založenou na systému souborů poskytující určitou ochranu proti selháním. Pro více specializovaných aplikací můžete použít `MQTTCLIENT_PERSISTENCE_USER`, který poskytuje rozhraní pro implementaci vašeho vlastního mechanizmu perzistence. Zda je perzistence vyžadována, je otázka návrhu aplikace. Další informace naleznete v tématu Perzistence zpráv.
- Výchozí démon TCP/IP démona pro MQTT je 1883. V tomto příkladu je výchozí adresa nastavena na `tcp://localhost:1883`.
- Dokud nezavoláte funkci `MQTTClient_connect` , neproběhne žádné zpracování zpráv.

8. Připojit klienta k démonu

```
if ((rc = MQTTClient_connect(client, &conn_opts)) != MQTTCLIENT_SUCCESS) {
    printf("Failed to connect, return code %d\n", rc);
    exit(-1);
}
```

   - Je volána funkce `MQTTClient_connect` , předává popisovač klienta a ukazatel na volby připojení jako argumenty.
   - Návratový kód z volání `MQTTClient_connect` se testuje, aby se ujistil, že požadavek na připojení je úspěšný.
   - Pokud se volání connect nezdaří, program skončí s kódem chyby -1.
   - Jakmile se aplikace připojí, může začít publikovat a odebírat.
   - Malá zpráva "keep-alive" se odesílá každých 20 sekund, aby se zabránilo zavření připojení TCP/IP. Tato volba je nastavena příkazem `conn_opts.keepAliveInterval`.
   - Relace se spustí bez kontroly dokončení průběžných zpráv, které zbývají z předchozího připojení, protože *conn_opts.cleansession* je nastaveno na true. Další informace najdete v tématu Vyčistit relace.
   - Pro připojení se nevytvoří žádná poslední zpráva a zpráva o potvrzení. Další podrobnosti naleznete v tématu Poslední podrobnosti a potvrzení

9. Přihlaste se k odběru tématu.

```
MQTTClient_subscribe(client, TOPIC, QOS);
```

   - Pomocí funkce produktu `MQTTClient_subscribe` se přihlašte k odběru klientské aplikace u vybraného tématu. Název tématu může obsahovat zástupné znaky. Další informace naleznete v tématu “Řetězce témat a filtry témat v klientech MQTT” na stránce 525.
   - Nastavení QoS určuje maximální kvalitu služby, která se použije na zprávy odeslané tomuto odběrateli. Server odešle zprávy na nižší hodnotu tohoto nastavení a nastavení QoS pro původní zprávu.
   - Tato funkce vrací kód chyby, který může být testován pro správné dokončení v produkčním kódu.

10. Počkejte ve smyčce, dokud uživatel nevstoupí do znaku 'Q' z klávesnice.

```
do {
    ch = getchar();
    } while(ch!='Q' && ch != 'q');
```

    Program nyní čeká na příchod zpráv. V tomto příkladu se všechna zpracování zpráv provádí ve funkci zpětného volání `MQTTClient_messageArrived`. Další informace naleznete v tématu “Příjem zpráv” na stránce 505.

11. Odpojte klienta od démona.

```
MQTTClient_disconnect(client, 10000);
```

- Klient se odpojí od serveru a čeká na všechny funkce zpětného volání (v tomto příkladu se nepoužívá), aby se dokončilo zpracování inflatézních zpráv.
- Druhý argument určuje časový limit uvedení do klidového stavu v milisekundách. Tento příklad čeká až 10 sekund na dokončení jakékoli jiné práce, kterou musí provést před odpojením.
- Tato funkce vrací kód chyby, který může být testován pro správné dokončení v produkčním kódu.

12. Uvolnění paměti použité klientem a ukončení programu.

```
MQTTClient_destroy(&client);
}
```

Příjem zpráv

Informace o této úloze

Když zprávy dorazí ze serveru, spustí se funkce `MQTTClient_messageArrived` . Následující kroky popisují kód.

Postup

1. Spusťte definici funkce zpětného volání. Tato definice musí odpovídat šabloně funkce `MQTTClient_messageArrived` .

```
int msgarrvd(void *context, char *topicName, int topicLen, MQTTClient_message *message) {
```

   - `context` poskytuje přístup k kontextu předanému do knihovny klienta, když byla volána funkce `MQTTClient_setCallbacks` . Tato funkce se v příkladu nepoužívá.
   - `topicName` je ukazatel na téma, do kterého je publikovaná zpráva publikována. Pokud jste přihlášeni k odběru pomocí zástupných znaků, tento parametr identifikuje specifické téma, které se používá pro danou zprávu.
   - `topicLen` je délka řetězce tématu. Tato volba je k dispozici pro uživatele, kteří musí v řetězcích témat vložit hodnotu NULL.
   - `message` je ukazatel na strukturu `MQTTClient_message` obsahující informační obsah zprávy a atributy.

2. Definujte použité lokální proměnné.

```
int i;
char* payloadptr;
```

   Tyto proměnné se používají v příkladu k vytištění informačního obsahu tím, že jej iteruje nad ním.

3. Vytisknout zprávu, zobrazit téma a informační obsah zprávy

```
printf("Message arrived\n");
printf("     topic: %s\n",topicName);
printf("   message: ");
payloadptr = message->payload;
for(i=0; i<message->payloadlen; i++){
    putchar(*payloadptr++);
    }
putchar('\n');
```

   - Tento příklad předpokládá, že přijatý informační obsah je posloupnost tisknutelných znaků.
   - Informační obsah MQTT je pole bajtů. Žádost je odpovědná za interpretaci jejich významu.

4. Uvolněte paměť používanou k ukládání zprávy.

```
MQTTClient_freeMessage(&message);
MQTTClient_free(topicName);
```

   - V tomto příkladu se všechna zpracování zpráv provádí ve funkci zpětného volání.

- Ujistěte se, že funkce zpětného volání jsou krátké, a vraťte řízení svému volajícímu podprocesu co nejdříve.
- Ukazatel zprávy je předán ke zpracování v hlavní části programu.
- Hlavní program musí uvolnit paměť, kterou používá zpráva při dokončení zpracování. `MQTTClient_freeMessage()` je užitečná funkce, která vrací dva paměťové bloky používané k uchování struktury *MQTTClient_message* a informačního obsahu zprávy zpět do systému. Paměť alokovaná pro `topicName` musí být uvolněna odděleně, jak je zobrazeno.

5. Vrátit hodnotu true, když zpětné volání bylo úspěšně zpracováno se zprávou

```
    return 1;
}
```

- Navrácení skutečné hodnoty znamená, že knihovna klienta může zpracovat zprávu jako úspěšně doručenou.
- Pokud funkce zpětného volání nemůže zprávu správně zpracovat, vrátí se nepravdivá hodnota. Například, pokud zpětné volání vkládá zprávy do fronty, aby se hlavní program zpracoval a fronta je plná, je vhodné vrátit nepravdivou hodnotu.
- Pro zprávy QoS1 a QoS2 vrácení nepravdivé hodnoty označuje, že zpráva nebyla doručena a další pokusy o její doručení byly provedeny.

Příklad kódu

```
#include "stdio.h"
#include "stdlib.h"
#include "MQTTClient.h"
#include "settings.h"
#include "callback.h"

int main(int argc, char* argv[]) {
    MQTTClient client;
    MQTTClient_connectOptions conn_opts = MQTTClient_connectOptions_initializer;
    int rc;
    int ch;

    MQTTClient_create(&client, ADDRESS, CLIENTID, MQTTCLIENT_PERSISTENCE_NONE, NULL);

    MQTTClient_setCallbacks(client, NULL, connlost, msgarrvd, delivered);

    if ((rc = MQTTClient_connect(client, &conn_opts)) != MQTTCLIENT_SUCCESS) {
        printf("Failed to connect, return code %d\n", rc);
        exit(-1);
        }
    printf("Subscribing to topic %s\nfor client %s using QoS%d\n\n"
           "Press Q<Enter> to quit\n\n", TOPIC, CLIENTID, QOS);

    MQTTClient_subscribe(client, TOPIC, QOS);
    do {
        ch = getchar();
        } while(ch!='Q' && ch != 'q');
    MQTTClient_disconnect(client, 10000);
    MQTTClient_destroy(&client);
}
```

Obrázek 117. `subscriber.c`

```
#include "MQTTClient.h"

volatile MQTTClient_deliveryToken deliveredtoken;

void delivered(void *context, MQTTClient_deliveryToken dt) {
    printf("Message with token value %d delivery confirmed\n", dt);
    deliveredtoken = dt;
}

int msgarrvd(void *context, char *topicName, int topicLen, MQTTClient_message *message) {
    int i;
    char* payloadptr;

    printf("Message arrived\n");
    printf("     topic: %s\n", topicName);
    printf("   message: ");

    payloadptr = message->payload;
    for(i=0; i<message->payloadlen; i++) {
        putchar(*payloadptr++);
        }
    putchar('\n');
    MQTTClient_freeMessage(&message);
    MQTTClient_free(topicName);
    return 1;
}
void connlost(void *context, char *cause) {
    printf("\nConnection lost\n");
    printf("     cause: %s\n", cause);
}
```

Obrázek 118. `callback.h`

```
#define ADDRESS    "tcp://localhost:1883"
#define CLIENTID   "ExampleClientSub"
#define TOPIC      "MQTT Example"
#define PAYLOAD    "Hello World!"
#define QOS        1
#define TIMEOUT    10000L
```

Obrázek 119. `settings.h`

Koncepty programování klienta MQTT

Koncepty popsané v tomto oddílu vám pomohou porozumět knihovnám klienta Java, JavaScript a C pro verzi 3.1 produktu MQTT protocol. Koncepty doplňují dokumentaci k rozhraní API doprovázející knihovny klienta.

`com.ibm.micro.client.mqttv3` obsahuje třídy, které poskytují veřejné metody pro knihovny klienta pro protokol MQTT verze 3.1 . Verze balíku produktu `com.ibm.micro.client.mqttv3` a doprovodné balíky, které implementují protokol pro produkt Java SE a ME, se poskytují spolu s instalací produktu IBM WebSphere MQ Telemetry. Chcete-li získat nejnovější verzi knihoven klienta MQTT (Java, JavaScript a chcete-li zobrazit nebo stáhnout dokumentaci k rozhraní API, prohlédněte si téma "Odkaz na programování klienta MQTT".

Chcete-li vyvinout a spustit klienta MQTT , musíte tyto balíky zkopírovat nebo nainstalovat na klientské zařízení. Není třeba instalovat oddělené běhové prostředí klienta.

Podmínky licencování pro klienty jsou přidruženy k serveru, ke kterému připojujete klienty.

Klientské knihovny produktu MQTT jsou referenční implementace verze 3.1 produktu MQTT protocol. Můžete implementovat vlastní klienty v různých jazycích vhodných pro různé platformy zařízení. Viz MQ Telemetry Transport format and protocol.

Dokumentace k rozhraním API nečiní žádné předpoklady o tom, ke kterému serveru MQTT je klient připojen. Chování klienta se může mírně lišit, pokud je připojeno k různým serverům. Následující popisy popisují chování klienta při připojení ke službě telemetrie IBM WebSphere MQ .

Zpětná volání a synchronizace v klientských aplikacích MQTT

Programovací model klienta produktu MQTT rozsáhle používá podprocesy. Podprocesy oddělující aplikaci klienta MQTT , stejně jako mohou, z prodlev při přenosu zpráv ze serveru a ze serveru. Publikování, tokeny doručení a ztracené události připojení se doručují do metod ve třídě zpětného volání, která implementuje `MqttCallback`.

Zpětná volání

Rozhraní `MqttCallback` má tři metody zpětného volání, viz příklad implementace v souboru Callback.java.

`connectionLost(java.lang.Throwable cause)`

`connectionLost` je volán, když komunikační chyba vede k uvolnění spojení. Nazývá se také tehdy, když server po navázání spojení přeruší spojení v důsledku chyby na serveru. Chyby serveru se protokolují do protokolu chyb správce front. Server zruší spojení s klientem a klient zavolá `MqttCallback.connectionLost`.

Jediné vzdálené chyby, vyvolané jako výjimky ze stejného podprocesu jako klientská aplikace, jsou výjimky z produktu `MqttClient.connect`. Chyby zjištěné serverem po zavedení připojení jsou nahlášeny zpět do metody zpětného volání `MqttCallback.connectionLost` jako `throwables`.

Typické chyby serveru, které vedou k chybě `connectionLost` , jsou chyby autorizace. Server telemetrie se například pokusí publikovat na téma jménem klienta, který není autorizován k publikování v rámci daného tématu. Vše, co má za následek vrácení kódu podmínky produktu `MQCC_FAIL` do serveru telemetrie, může vést k tomu, že připojení bude zrušeno.

`deliveryComplete(MqttDeliveryToken token)`

`deliveryComplete` je volán klientem MQTT k předání tokenu doručení klientské aplikaci; viz “Tokeny doručení” na stránce 513. Při použití tokenu doručení může zpětné volání přistupovat k publikované zprávě s použitím metody `token.getMessage`.

Když zpětné volání aplikace vrátí řízení do klienta MQTT po volání metody `deliveryComplete` , doručení je dokončeno. Dokud nebude dokončeno doručení, zprávy s produktem `QoS` 1 nebo 2 jsou zachovány podle třídy perzistence.

Volání do produktu `deliveryComplete` je bodem synchronizace mezi aplikací a třídou perzistence. Metoda `deliveryComplete` se nikdy nevolá dvakrát pro stejnou zprávu.

Když se zpětné volání aplikace vrátí z produktu `deliveryComplete` na klienta MQTT , klient zavolá příkaz `MqttClientPersistence.remove` pro zprávy s hodnotou `QoS` 1 nebo 2. Příkaz `MqttClientPersistence.remove` odstraní lokálně uloženou kopii publikované zprávy.

Z pohledu zpracování transakce je volání na produkt `deliveryComplete` jednofázovou transakcí, která potvrzuje doručení. Pokud zpracování selže během zpětného volání, při restartu klienta `MqttClientPersistence.remove` se znovu volá a odstraní lokální kopie publikované zprávy. Zpětné volání se nevolá znovu. Pokud používáte zpětné volání k uložení protokolu doručených zpráv, nemůžete synchronizovat protokol s klientem MQTT . Chcete-li uložit protokol spolehlivě, aktualizujte protokol ve třídě `MqttClientPersistence` .

Na token doručení a na zprávu odkazuje hlavní aplikační podproces a klient produktu MQTT . Klient MQTT zruší odkaz na objekt `MqttMessage` při dokončení doručení a objekt tokenu doručení, když se klient odpojí. Objekt `MqttMessage` může být vyčištěn odpad po dokončení doručení, pokud aplikace klienta zruší odkaz na něj. Po odpojení relace může být token doručení odebrán.

Po publikování zprávy můžete získat atributy `MqttDeliveryToken` a `MqttMessage` po publikování zprávy. Pokusíte-li se nastavit libovolný atribut `MqttMessage` poté, co byla zpráva publikována, výsledek není definován.

Klient MQTT pokračuje v zpracování potvrzení doručení, pokud se klient znovu připojí k předchozí relaci se stejným parametrem `ClientIdentifier`; viz “Vyčistit relace” na stránce 511. Aplikace klienta MQTT musí nastavit `MqttClient.CleanSession` na `false` pro předchozí relaci a nastavit ji na `false` v nové relaci. Klient produktu MQTT vytváří nové tokeny doručení a objekty zpráv v nové relaci pro nevyřízené doručení. Zotavuje objekty pomocí třídy `MqttClientPersistence` . Pokud má klient aplikace stále odkazy na staré tokeny doručení

a zprávy, vyhodnoťte je. Zpětné volání aplikace se volá v nové relaci pro všechny doručení zahájené v předchozí relaci a dokončené v této relaci.

Zpětné volání aplikace se volá po připojení klienta aplikace, když je dokončeno nevyřízené doručení. Než se klient aplikace připojí, může nevyřízené doručení načíst pomocí metody `MqttClient.getPendingDeliveryTokens` .

Všimněte si, že aplikace klienta původně vytvořila objekt zprávy, který je publikován, a jeho bajtové pole informačního obsahu. Klient MQTT odkazuje na tyto objekty. Objekt zprávy vrácený tokenem doručení v metodě `token.getMessage` nemusí nutně znamenat stejný objekt zprávy vytvořený klientem. Pokud nová instance klienta MQTT znovu vytvoří token doručení, třída `MqttClientPersistence` znovu vytvoří objekt `MqttMessage` . Pro konzistenci `token.getMessage` vrátí hodnotu `null` , pokud `token.isCompleted` je `true`, bez ohledu na to, zda byl objekt zprávy vytvořen klientem aplikace nebo třídou `MqttClientPersistence` .

`messageArrived(MqttTopic topic, MqttMessage message)`

`messageArrived` je volán, když je doručena publikace pro klienta, který odpovídá tématu odběru. `topic` je téma publikace, nikoli filtr odběru. Mohou se lišit, pokud filtr obsahuje zástupné znaky.

Pokud se téma shoduje s více odběry vytvořenými klientem, obdrží klient více kopií publikování. Pokud klient publikuje do tématu, které se také přihlásí k odběru, obdrží kopii své vlastní publikace.

Je-li zpráva odeslána s `QoS` z `1` nebo `2`, zpráva je uložena třídou `MqttClientPersistence` před voláním klienta MQTT `messageArrived`. Příkaz `messageArrived` se chová jako `deliveryComplete`: volá se pouze jednou pro publikování a lokální kopie této publikace je odstraněna produktem `MqttClientPersistence.remove` , když se příkaz `messageArrived` vrátí klientovi MQTT . Klient MQTT zruší své odkazy na téma a zprávu, když se `messageArrived` vrátí klientovi MQTT . Objekty tématu a zprávy jsou shromažďovány v rámci uvolňování paměti, pokud klient aplikace neuchoval odkaz na objekty.

Zpětná volání, rozdělování na podprocesy a synchronizace aplikací klienta

Klient MQTT volá metodu zpětného volání na samostatný podproces do hlavního podprocesu aplikace. Klientská aplikace 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 umožňuje snadno aktualizovat objekt, který obsahuje informace o tom, které publikace byly doručeny. Jedna instance serveru `MqttCallback.deliveryComplete` se spustí v daném okamžiku, a proto je bezpečné aktualizovat záznam, aniž by došlo k další synchronizaci. Je také tomu tak, že v daném okamžiku dorazí pouze jedna publikace. Váš kód v metodě `messageArrived` může aktualizovat objekt, aniž by se synchronizoval. Pokud odkazujete na záznam nebo objekt, který se právě aktualizuje, v jiném podprocesu synchronizujte záznam nebo objekt.

Token doručení poskytuje mechanizmus synchronizace mezi hlavním podprocesem aplikace a doručením publikace. Metoda `token.waitForCompletion` čeká, dokud nebude dokončeno doručení určité publikace, nebo dokud nevyprší volitelný časový limit. Produkt `token.waitForCompletion` můžete použít v několika jednoduchých způsobech ke zpracování jedné publikace v daném okamžiku:

1. Chcete-li klienta aplikace pozastavit až do dokončení doručení publikace, prohlédněte si téma Obrázek 88 na stránce 464.
2. Proveďte synchronizaci s metodou `MqttCallback.deliveryComplete` . Pouze když se `MqttCallback.deliveryComplete` vrátí na klienta MQTT , obnoví se `token.waitForCompletion` . Pomocí tohoto mechanizmu můžete synchronizovat spouštěný kód v produktu `MqttCallback.deliveryComplete` před spuštěním kódu v hlavním podprocesu aplikace.

Co když jste chtěli publikovat bez čekání na doručení každé publikace, ale chcete potvrdit, kdy byly všechny publikace doručeny? Pokud publikujete na jednom podprocesu, poslední publikování, které má být odesláno, je také poslední, které má být doručeno.

Synchronizace požadavků odeslaných na server

Tabulka 70 na stránce 510 popisuje metody v klientovi MQTT Java, které odesílají požadavek na server. Pokud aplikační klient nenastaví neomezenou časový limit, klient nikdy na server nečeká nekonečně dlouhou dobu. Pokud se klient zablokuje, je to buď problém programování aplikace, nebo defekt v klientovi MQTT .

Tabulka 70. Chování synchronizace metod, které vedou k požadavkům na server

Metoda	Synchronizace	Interval časového limitu
`MqttClient.Connect`	Čeká na navázání spojení se serverem.	Výchozí hodnota je 30 sekund, nebo jak je nastaveno parametrem, pak vyvolá výjimku.
`MqttClient.Disconnect`	Čeká na dokončení práce klienta MQTT a jeho odpojení od relace TCP/IP.	
`MqttClient.Subscribe`	Čeká na dokončení metody odběru nebo odběru UnSubscribe .	
`MqttClient.UnSubscribe`		
`MqttClient.Publish`	Po předání požadavku klientovi produktu MQTT se okamžitě vrátí k podprocesu aplikace.	Není.
`MqttDeliveryToken.waitForCompletion`	Čeká na vrácení doručovacího tokenu.	Neomezené, nebo 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" na příjmu publikací. Stav relace také zahrnuje odběry vytvořené klientem MQTT. Můžete zvolit spuštění klienta MQTT s nebo bez udržování informací o stavu mezi relacemi. Změňte režim čisté relace nastavením `MqttConnectOptions.cleanSession` před připojením.

Identifikátor klienta

Tokeny doručení

Datum poslední vůle a potvrzení
Pokud došlo k neočekávanému ukončení připojení klienta MQTT , můžete produkt WebSphere MQ Telemetry nakonfigurovat tak, aby odeslal publikaci "last will and testament". Předefinujte obsah publikace a téma, kterému jej chcete odeslat. "Poslední vůle a testament" je vlastnost připojení. Vytvořte jej před připojením klienta.

Perzistence zpráv v klientech produktu MQTT

Publikace
Publikace jsou instance `MqttMessage` , které jsou přidruženy k řetězci tématu. Klient produktu MQTT může vytvářet publikace k odesílání do produktu IBM WebSphere MQa k odběru témat v produktu IBM WebSphere MQ MQ za přihlášení k odběru publikací.

Kvality služby poskytované klientem MQTT
Klient produktu MQTT poskytuje tři úrovně kvality služby pro dodávání publikací do produktu WebSphere MQ a do klienta MQTT : "maximálně jednou", "alespoň jednou" a "přesně jednou". Pokud klient produktu MQTT odešle požadavek na produkt WebSphere MQ pro vytvoření odběru, odešle se požadavek s kvalitou služby "alespoň jednou".

Zachovaná publikování a klienti MQTT
Pokud vytvoříte odběr tématu, které má zachované publikování, bude vám okamžitě předáno nejnovější zachované publikování v tématu.

Odběry
Vytvoření odběrů pro registraci zájmu v tématech publikování pomocí filtru témat. Klient může vytvořit více odběrů nebo odběr obsahující filtr témat, který používá zástupné znaky, a zaregistrovat tak zájem o více témat. Publikace na témata, která odpovídají filtrům, se posílají na klienta. Odběry mohou zůstat aktivní, když je klient odpojen. Publikace se odesílají klientovi, jakmile se znovu připojí.

Řetězce témat a filtry témat v klientech MQTT
Řetězce témat a filtry témat se používají pro publikování a odběr. Syntaxe řetězců témat a filtrů v klientech produktu MQTT je z velké části stejné jako řetězce témat v produktu IBM WebSphere 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" na příjmu publikací. Stav relace také zahrnuje odběry vytvořené klientem MQTT. Můžete zvolit spuštění klienta MQTT s nebo bez udržování informací o stavu mezi relacemi. Změňte režim čisté relace nastavením `MqttConnectOptions.cleanSession` před připojením.

Připojujete-li aplikaci klienta protokolu 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 jsou předchozí informace o relaci na klientovi a serveru vymazány. Je-li `cleanSession=false` předchozí relace obnovena. Neexistuje-li žádná předchozí relace, bude spuštěna nová relace.

Poznámka: Administrátor produktu WebSphere MQ může vynutit zavření otevřené relace a odstranění všech informací o relaci. Pokud klient znovu otevře relaci s produktem `cleanSession=false`, spustí se nová relace.

Publikace

Pokud použijete výchozí `MqttConnectOptions`nebo nastavíte `MqttConnectOptions.cleanSession` na `true` před připojením klienta, všechny nevyřízené doručení publikování pro klienta se odeberou, když se klient připojí.

Nastavení čisté relace nemá žádný vliv na publikace odeslané s produktem `QoS=0`. Pro `QoS=1` a `QoS=2`může použití `cleanSession=true` vést ke ztrátě publikování.

Odběry

Používáte-li výchozí hodnotu `MqttConnectOptions` nebo před připojením klienta nastavíte `MqttConnectOptions.cleanSession` na hodnotu `true`, budou všechny staré odběry klienta při připojení odebrány. Všechny nové odběry, které klient během relace vytvoří, jsou při odpojení odebrány.

Pokud před připojením nastavíte `MqttConnectOptions.cleanSession` na hodnotu `false`, budou všechny vytvořené odběry klienta přidány ke všem odběrům, které existovaly pro klienta 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 je třeba nastavit režim `cleanSession`; režim trvá pro celou relaci. Chcete-li změnit jeho nastavení, je třeba klienta odpojit a znovu připojit. Pokud změníte režimy z používání `cleanSession=false` na `cleanSession=true`, budou všechny předchozí odběry pro klienta a všechny publikace, které nebyly přijaty, vyřazeny.

Související pojmy
Zpětná volání a synchronizace v klientských aplikacích MQTT
Programovací model klienta produktu MQTT rozsáhle používá podprocesy. Podprocesy oddělující aplikaci klienta MQTT , stejně jako mohou, z prodlev při přenosu zpráv ze serveru a ze serveru. Publikování, tokeny

doručení a ztracené události připojení se doručují do metod ve třídě zpětného volání, která implementuje `MqttCallback`.

Identifikátor klienta

Tokeny doručení

Datum poslední vůle a potvrzení
Pokud došlo k neočekávanému ukončení připojení klienta MQTT , můžete produkt WebSphere MQ Telemetry nakonfigurovat tak, aby odeslal publikaci "last will and testament". Předefinujte obsah publikace a téma, kterému jej chcete odeslat. "Poslední vůle a testament" je vlastnost připojení. Vytvořte jej před připojením klienta.

Perzistence zpráv v klientech produktu MQTT

Publikace
Publikace jsou instance `MqttMessage` , které jsou přidruženy k řetězci tématu. Klient produktu MQTT může vytvářet publikace k odesílání do produktu IBM WebSphere MQa k odběru témat v produktu IBM WebSphere MQ MQ za přihlášení k odběru publikací.

Kvality služby poskytované klientem MQTT
Klient produktu MQTT poskytuje tři úrovně kvality služby pro dodávání publikací do produktu WebSphere MQ a do klienta MQTT : "maximálně jednou", "alespoň jednou" a "přesně jednou". Pokud klient produktu MQTT odešle požadavek na produkt WebSphere MQ pro vytvoření odběru, odešle se požadavek s kvalitou služby "alespoň jednou".

Zachovaná publikování a klienti MQTT
Pokud vytvoříte odběr tématu, které má zachované publikování, bude vám okamžitě předáno nejnovější zachované publikování v tématu.

Odběry
Vytvoření odběrů pro registraci zájmu v tématech publikování pomocí filtru témat. Klient může vytvořit více odběrů nebo odběr obsahující filtr témat, který používá zástupné znaky, a zaregistrovat tak zájem o více témat. Publikace na témata, která odpovídají filtrům, se posílají na klienta. Odběry mohou zůstat aktivní, když je klient odpojen. Publikace se odesílají klientovi, jakmile se znovu připojí.

Řetězce témat a filtry témat v klientech MQTT
Řetězce témat a filtry témat se používají pro publikování a odběr. Syntaxe řetězců témat a filtrů v klientech produktu MQTT je z velké části stejné jako řetězce témat v produktu IBM WebSphere MQ.

Identifikátor klienta

Identifikátor klienta je 23bajtový řetězec, který identifikuje klienta MQTT . Každý identifikátor musí být jedinečný, tedy platný ve stejnou dobu pouze pro jediného připojeného klienta. Identifikátor musí obsahovat pouze znaky platné v názvu správce front. V rámci těchto omezení budete moci používat libovolný identifikační řetězec. Je důležité mít proceduru pro přidělování identifikátorů klienta a prostředků ke konfiguraci klienta se zvoleným identifikátorem.

Identifikátor klienta se používá při administraci systému MQTT . S potencionálně stovkami tisíc klientů, kteří mají spravovat, musíte být schopni rychle identifikovat konkrétního klienta. Předpokládejme například, že zařízení má nefunkčnost, a vy budete informováni o tom, že zákazník zazvoní na help desk. Jak zákazník identifikuje zařízení a jak korelovat s identifikací na serveru, který je obvykle připojen ke klientovi? Je třeba se poradit s databází, která mapuje každé zařízení na identifikátor klienta a na server? Identifikujete název zařízení, ke kterému serveru je připojen? Když procházíte přes připojení klienta MQTT , každé připojení je označeno identifikátorem klienta. Potřebujete vyhledat tabulku, chcete-li mapovat identifikátor klienta na fyzické zařízení?

Identifikujete identifikátor klienta určité zařízení, uživatele nebo aplikaci spuštěnou na klientovi? Pokud zákazník nahradí vadné zařízení novým zařízením, má nové zařízení stejný identifikátor jako staré zařízení? Přidělíte nový identifikátor? Změníte-li fyzické zařízení, ale zachovejte stejný identifikátor, význačné publikace a aktivní odběry se automaticky přenesou na nové zařízení.

Jak se ujistíte, že identifikátory klientů jsou jedinečné? Stejně jako systém pro generování jedinečných identifikátorů, musíte mít spolehlivý proces pro nastavení identifikátoru na klientovi. Možná klientské zařízení je "black-box", bez uživatelského rozhraní. Vyráběte zařízení s identifikátorem klienta-například

pomocí jeho MAC adresy? Nebo máte instalaci softwaru a proces konfigurace, který konfiguruje zařízení před aktivací zařízení?

Můžete vytvořit identifikátor klienta z 48bitové adresy MAC zařízení, abyste zachovali identifikátor krátký a jedinečný. Pokud velikost přenosu není kritickým problémem, můžete použít zbývajících 17 bajtů, abyste mohli snáze spravovat adresu.

Související pojmy

Zpětná volání a synchronizace v klientských aplikacích MQTT
Programovací model klienta produktu MQTT rozsáhle používá podprocesy. Podprocesy oddělující aplikaci klienta MQTT , stejně jako mohou, z prodlev při přenosu zpráv ze serveru a ze serveru. Publikování, tokeny doručení a ztracené události připojení se doručují do metod 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" na příjmu publikací. Stav relace také zahrnuje odběry vytvořené klientem MQTT. Můžete zvolit spuštění klienta MQTT s nebo bez udržování informací o stavu mezi relacemi. Změňte režim čisté relace nastavením `MqttConnectOptions.cleanSession` před připojením.

Tokeny doručení

Datum poslední vůle a potvrzení
Pokud došlo k neočekávanému ukončení připojení klienta MQTT , můžete produkt WebSphere MQ Telemetry nakonfigurovat tak, aby odeslal publikaci "last will and testament". Předefinujte obsah publikace a téma, kterému jej chcete odeslat. "Poslední vůle a testament" je vlastnost připojení. Vytvořte jej před připojením klienta.

Perzistence zpráv v klientech produktu MQTT

Publikace
Publikace jsou instance `MqttMessage` , které jsou přidruženy k řetězci tématu. Klient produktu MQTT může vytvářet publikace k odesílání do produktu IBM WebSphere MQa k odběru témat v produktu IBM WebSphere MQ MQ za přihlášení k odběru publikací.

Kvality služby poskytované klientem MQTT
Klient produktu MQTT poskytuje tři úrovně kvality služby pro dodávání publikací do produktu WebSphere MQ a do klienta MQTT : "maximálně jednou", "alespoň jednou" a "přesně jednou". Pokud klient produktu MQTT odešle požadavek na produkt WebSphere MQ pro vytvoření odběru, odešle se požadavek s kvalitou služby "alespoň jednou".

Zachovaná publikování a klienti MQTT
Pokud vytvoříte odběr tématu, které má zachované publikování, bude vám okamžitě předáno nejnovější zachované publikování v tématu.

Odběry
Vytvoření odběrů pro registraci zájmu v tématech publikování pomocí filtru témat. Klient může vytvořit více odběrů nebo odběr obsahující filtr témat, který používá zástupné znaky, a zaregistrovat tak zájem o více témat. Publikace na témata, která odpovídají filtrům, se posílají na klienta. Odběry mohou zůstat aktivní, když je klient odpojen. Publikace se odesílají klientovi, jakmile se znovu připojí.

Řetězce témat a filtry témat v klientech MQTT
Řetězce témat a filtry témat se používají pro publikování a odběr. Syntaxe řetězců témat a filtrů v klientech produktu MQTT je z velké části stejné jako řetězce témat v produktu IBM WebSphere MQ.

Tokeny doručení

Když klient publikuje na téma nový token doručení, dojde k vytvoření nového tokenu doručení. Použijte token doručení k monitorování doručení publikování nebo k blokování klientské aplikace, dokud nebude doručení dokončeno.

Token je objekt `MqttDeliveryToken` . Vytvoří se voláním metody MqttTopic.publish () a zachová ji klient produktu MQTT , dokud nebude relace klienta odpojena a doručení je dokončeno.

Normální použití tokenu je zkontrolovat, zda je doručení dokončeno. Zablokujte aplikaci klienta, dokud se doručení nedokončí s použitím vráceného tokenu k volání `token.waitForCompletion`. Případně poskytněte obslužnou rutinu `MqttCallBack` . Pokud klient produktu MQTT přijal všechna potvrzení, která očekává jako část doručení publikace, zavolá příkaz `MqttCallBack.deliveryComplete` předáním tokenu doručení jako parametru.

Dokud nebude dodávka dokončena, můžete ji zkontrolovat pomocí vráceného tokenu doručení voláním produktu `token.getMessage`.

Dokončené dodávky

Dokončení dodávek je asynchronní a závisí na kvalitě služby přidružené k publikování.

Nejvíce jednou

`QoS=0`
Doručení je dokončeno okamžitě po návratu z produktu `MqttTopic.publish`. `MqttCallback.deliveryComplete` se volá okamžitě.

Nejméně jednou

`QoS=1`
Doručení je dokončeno, když bylo od správce front přijato potvrzení o publikování. `MqttCallback.deliveryComplete` se volá, když je přijato potvrzení. Zpráva může být doručena více než jednou dříve, než se zavolá `MqttCallback.deliveryComplete` , pokud jsou komunikace pomalé nebo nespolehlivé.

Přesně jednou

`QoS=2`
Doručení je dokončeno, když klient obdrží zprávu o dokončení, že publikování bylo publikováno odběratelům. `MqttCallback.deliveryComplete` se volá, jakmile se přijme zpráva o publikování. Nečeká na zprávu o dokončení.

Za výjimečných okolností se nemusí klientská aplikace vrátit ke klientovi MQTT z produktu `MqttCallback.deliveryComplete` normálně. Víte, že dodávka byla dokončena, protože byl volán `MqttCallback.deliveryComplete` . Pokud klient restartuje stejnou relaci, produkt `MqttCallback.deliveryComplete` se znovu nezavolá.

Neúplné dodávky

Pokud není doručení dokončeno po odpojení relace klienta, můžete klienta znovu připojit a dokončit doručení. Doručování zprávy můžete dokončit pouze v případě, že byla zpráva publikována v relaci s atributem `MqttConnectionOptions` nastaveným na hodnotu `false`.

Vytvořte klienta pomocí stejného identifikátoru klienta a adresy serveru a potom se připojte znovu, nastavte atribut `cleanSession` `MqttConnectionOptions` na hodnotu `false` znovu. Pokud jste nastavili `cleanSession` na `true`, nevyřízené tokeny doručení budou zahozeny.

Můžete zkontrolovat, zda neexistují nějaké nevyřízené doručení voláním produktu `MqttClient.getPendingDeliveryTokens`. Před připojením klienta můžete volat `MqttClient.getPendingDeliveryTokens` .

Související pojmy

Zpětná volání a synchronizace v klientských aplikacích MQTT
Programovací model klienta produktu MQTT rozsáhle používá podprocesy. Podprocesy oddělující aplikaci klienta MQTT , stejně jako mohou, z prodlev při přenosu zpráv ze serveru a ze serveru. Publikování, tokeny doručení a ztracené události připojení se doručují do metod 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" na příjmu publikací. Stav relace také zahrnuje odběry vytvořené klientem MQTT. Můžete zvolit spuštění klienta

MQTT s nebo bez udržování informací o stavu mezi relacemi. Změňte režim čisté relace nastavením `MqttConnectOptions.cleanSession` před připojením.

Identifikátor klienta

Datum poslední vůle a potvrzení
Pokud došlo k neočekávanému ukončení připojení klienta MQTT , můžete produkt WebSphere MQ Telemetry nakonfigurovat tak, aby odeslal publikaci "last will and testament". Předefinujte obsah publikace a téma, kterému jej chcete odeslat. "Poslední vůle a testament" je vlastnost připojení. Vytvořte jej před připojením klienta.

Perzistence zpráv v klientech produktu MQTT

Publikace
Publikace jsou instance `MqttMessage` , které jsou přidruženy k řetězci tématu. Klient produktu MQTT může vytvářet publikace k odesílání do produktu IBM WebSphere MQa k odběru témat v produktu IBM WebSphere MQ MQ za přihlášení k odběru publikací.

Kvality služby poskytované klientem MQTT
Klient produktu MQTT poskytuje tři úrovně kvality služby pro dodávání publikací do produktu WebSphere MQ a do klienta MQTT : "maximálně jednou", "alespoň jednou" a "přesně jednou". Pokud klient produktu MQTT odešle požadavek na produkt WebSphere MQ pro vytvoření odběru, odešle se požadavek s kvalitou služby "alespoň jednou".

Zachovaná publikování a klienti MQTT
Pokud vytvoříte odběr tématu, které má zachované publikování, bude vám okamžitě předáno nejnovější zachované publikování v tématu.

Odběry
Vytvoření odběrů pro registraci zájmu v tématech publikování pomocí filtru témat. Klient může vytvořit více odběrů nebo odběr obsahující filtr témat, který používá zástupné znaky, a zaregistrovat tak zájem o více témat. Publikace na témata, která odpovídají filtrům, se posílají na klienta. Odběry mohou zůstat aktivní, když je klient odpojen. Publikace se odesílají klientovi, jakmile se znovu připojí.

Řetězce témat a filtry témat v klientech MQTT
Řetězce témat a filtry témat se používají pro publikování a odběr. Syntaxe řetězců témat a filtrů v klientech produktu MQTT je z velké části stejné jako řetězce témat v produktu IBM WebSphere MQ.

Datum poslední vůle a potvrzení

Pokud došlo k neočekávanému ukončení připojení klienta MQTT , můžete produkt WebSphere MQ Telemetry nakonfigurovat tak, aby odeslal publikaci "last will and testament". Předefinujte obsah publikace a téma, kterému jej chcete odeslat. "Poslední vůle a testament" je vlastnost připojení. Vytvořte jej před připojením klienta.

Vytvořte téma pro poslední vůli a testament. Můžete vytvořit téma jako např. `MQTTManagement/Connections/`*`server URI`*`/`*`client identifer`*`/Lost`.

Nastavte "poslední vůli a testament" pomocí metody `MqttConnectionOptions.setWill(MqttTopic lastWillTopic, byte [] lastWillPayload, int lastWillQos, boolean lastWillRetained)` .

Zvažte vytvoření časového razítka ve zprávě `lastWillPayload` . Zahrnout další informace o klientovi, které pomáhají při identifikaci klienta a okolnosti připojení. Předejte objekt `MqttConnectionOptions` konstruktoru `MqttClient` .

Chcete-li zprávu trvalou v produktu WebSphere MQa zajistit doručení, nastavte hodnotu `lastWillQos` na hodnotu 1 nebo 2. Chcete-li zachovat informace o naposledy ztraceném připojení, nastavte parametr `lastWillRetained` na hodnotu `true`.

Publikování "poslední vůle a testament" je odesláno odběratelům, pokud připojení skončí neočekávaně. Je odeslán, pokud připojení skončí, aniž by klient volal metodu `MqttClient.disconnect` .

Chcete-li monitorovat připojení, doplním publikace "last will and testament" s dalšími publikacemi k záznamu připojení a programovaným odpojením.

Související pojmy

Zpětná volání a synchronizace v klientských aplikacích MQTT
Programovací model klienta produktu MQTT rozsáhle používá podprocesy. Podprocesy oddělující aplikaci klienta MQTT , stejně jako mohou, z prodlev při přenosu zpráv ze serveru a ze serveru. Publikování, tokeny doručení a ztracené události připojení se doručují do metod 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" na příjmu publikací. Stav relace také zahrnuje odběry vytvořené klientem MQTT. Můžete zvolit spuštění klienta MQTT s nebo bez udržování informací o stavu mezi relacemi. Změňte režim čisté relace nastavením `MqttConnectOptions.cleanSession` před připojením.

Identifikátor klienta

Tokeny doručení

Perzistence zpráv v klientech produktu MQTT

Publikace
Publikace jsou instance `MqttMessage` , které jsou přidruženy k řetězci tématu. Klient produktu MQTT může vytvářet publikace k odesílání do produktu IBM WebSphere MQa k odběru témat v produktu IBM WebSphere MQ MQ za přihlášení k odběru publikací.

Kvality služby poskytované klientem MQTT
Klient produktu MQTT poskytuje tři úrovně kvality služby pro dodávání publikací do produktu WebSphere MQ a do klienta MQTT : "maximálně jednou", "alespoň jednou" a "přesně jednou". Pokud klient produktu MQTT odešle požadavek na produkt WebSphere MQ pro vytvoření odběru, odešle se požadavek s kvalitou služby "alespoň jednou".

Zachovaná publikování a klienti MQTT
Pokud vytvoříte odběr tématu, které má zachované publikování, bude vám okamžitě předáno nejnovější zachované publikování v tématu.

Odběry
Vytvoření odběrů pro registraci zájmu v tématech publikování pomocí filtru témat. Klient může vytvořit více odběrů nebo odběr obsahující filtr témat, který používá zástupné znaky, a zaregistrovat tak zájem o více témat. Publikace na témata, která odpovídají filtrům, se posílají na klienta. Odběry mohou zůstat aktivní, když je klient odpojen. Publikace se odesílají klientovi, jakmile se znovu připojí.

Řetězce témat a filtry témat v klientech MQTT
Řetězce témat a filtry témat se používají pro publikování a odběr. Syntaxe řetězců témat a filtrů v klientech produktu MQTT je z velké části stejné jako řetězce témat v produktu IBM WebSphere MQ.

Perzistence zpráv v klientech produktu MQTT

Publikační zprávy jsou prováděny jako trvalé, pokud jsou odeslány s kvalitou služby alespoň jednou, nebo "přesně jednou". Na klientovi můžete implementovat vlastní mechanismus perzistence nebo použít výchozí mechanismus perzistence, který je dodáván s klientem. Perzistence funguje v obou směrech, pro publikování odeslané na klienta nebo z klienta.

V produktu MQTTmá perzistence zpráv dva aspekty, jak je zpráva přenášena a zda je zařazena do fronty v IBM MessageSight a IBM WebSphere MQ jako trvalá zpráva.

1. Perzistence zpráv klientských párů MQTT s kvalitou služeb. V závislosti na tom, jakou kvalitu služby vyberete pro zprávu, bude zpráva trvalá. Perzistence zpráv je nezbytná k implementaci požadované kvality služby.

   Určíte-li "maximálně jednou", `QoS=0`, klient zprávu zahodí, jakmile bude publikován. Dojde-li k selhání v předchozím zpracování zprávy, zpráva se neodešle znovu. I v případě, že klient zůstane aktivní, zpráva se znovu neodešle. Chování zpráv produktu `QoS=0` se shoduje s chováním IBM WebSphere MQ rychlých přechodných zpráv.

Je-li zpráva publikována klientem s QoS z 1 nebo 2, je vytrvalá. Zpráva je uložena lokálně a pouze vyřazena z klienta, když již není zapotřebí k záruce "alespoň jednou", QoS=1nebo "přesně jednou", QoS=2, doručení.

2. Je-li zpráva označena jako QoS 1 nebo 2, je zařazena do fronty v IBM MessageSight a IBM WebSphere MQ jako trvalá zpráva. Je-li označena jako QoS=0, pak je zařazena do fronty v IBM MessageSight a IBM WebSphere MQ jako přechodná zpráva. V produktu IBM WebSphere MQ jsou přechodné zprávy přenášeny mezi správci front "přesně jednou", pokud kanál zpráv nemá nastaven atribut NPMSPEED na hodnotu FAST.

Trvalá publikace je uložena v klientu, dokud ji nebude přijímat klientská aplikace. Pro produkt QoS=2je publikace vyřazena z klienta, když zpětné volání aplikace vrátí řízení. V případě QoS=1 může aplikace obdržet publikování znovu, pokud dojde k selhání. V případě systému QoS=0zpětné volání přijme publikování ne více než jednou. Pokud dojde k selhání nebo je-li klient odpojen v době publikování, nemusí být tato publikace zveřejněna.

Pokud se přihlásíte k odběru tématu, můžete snížit úroveň služeb QoS , se kterou odběratel přijímá zprávy, aby odpovídal schopnostem perzistence. Publikace, které jsou vytvořeny na vyšší verzi QoS , jsou odeslány s nejvyšší hodnotou QoS , kterou odběratel požadoval.

Ukládání zpráv

Implementace datového úložiště na malých zařízeních se velmi liší. Model dočasně ukládající trvalé zprávy v úložišti, který je spravován klientem MQTT , může být příliš pomalý, nebo požadovat příliš mnoho úložiště. V mobilních zařízeních může mobilní operační systém poskytovat službu úložiště, která je ideální pro zprávy produktu MQTT .

Klient MQTT má dvě rozhraní perzistence, aby byla zajištěna flexibilita při plnění omezení malých zařízení. Rozhraní definují operace, které jsou zapojeny do ukládání trvalých zpráv. Rozhraní jsou popsána v dokumentaci rozhraní API pro produkt Klient MQTT pro produkt Java. Pro odkazy na dokumentaci ke klientským rozhraním API pro knihovny klienta MQTT viz Odkaz na programování klienta MQTT. Rozhraní můžete implementovat tak, aby vyhovovala zařízení. Klient produktu MQTT spuštěný v prostředí Java SE má výchozí implementaci rozhraní, která ukládají trvalé zprávy v systému souborů. Používá balík `java.io` . Klient má také výchozí implementaci pro prostředí Java ME, `MqttDefaultMIDPPersistence`.

Třídy perzistence

`MqttClientPersistence`
Předejte instanci implementace produktu `MqttClientPersistence` na klienta produktu MQTT jako parametr konstruktoru `MqttClient` . Pokud vynecháte argument `MqttClientPersistence` z konstruktoru `MqttClient` , klient MQTT ukládá trvalé zprávy pomocí třídy `MqttDefaultFilePersistence` nebo `MqttDefaultMIDPPersistence`.

`MqttPersistable`
`MqttClientPersistence` získává a vkládá objekty `MqttPersistable` pomocí klíče úložiště. Musíte poskytnout implementaci produktu `MqttPersistable` , stejně jako implementaci produktu `MqttClientPersistence` , pokud nepoužíváte `MqttDefaultFilePersistence` nebo `MqttDefaultMIDPPersistence`.

`MqttDefaultFilePersistence`

Klient MQTT poskytuje třídu `MqttDefaultFilePersistence` . Pokud konkretizovat `MqttDefaultFilePersistence` v aplikaci klienta, můžete poskytnout adresář k ukládání trvalých zpráv jako parametru konstruktoru `MqttDefaultFilePersistence` .

Alternativně může klient MQTT vytvořit instanci `MqttDefaultFilePersistence` a umístit soubory do výchozího adresáře. Název adresáře je *`client identifier`*`-tcp` *`hostname portnumber`*. `"\", "\\", "/", ":"` a `" "` jsou odebrány z řetězce názvu adresáře.

Cesta k adresáři je hodnota vlastnosti systému `rcp.data`. Není-li parametr `rcp.data` nastaven, cesta je hodnota vlastnosti systému `usr.data`.

`rcp.data` je vlastnost přidružená k instalaci platformy OSGi nebo platformy Eclipse Rich Client Platform (RCP).

`usr.data` je adresář, ve kterém byl spuštěn příkaz jazyka Java, který spustil aplikaci.

`MqttDefaultMIDPPersistence`
`MqttDefaultMIDPPersistence` má výchozí konstruktor a žádné parametry. Používá balík produktu `javax.microedition.rms.RecordStore` k ukládání zpráv.

Související pojmy

Zpětná volání a synchronizace v klientských aplikacích MQTT
Programovací model klienta produktu MQTT rozsáhle používá podprocesy. Podprocesy oddělující aplikaci klienta MQTT , stejně jako mohou, z prodlev při přenosu zpráv ze serveru a ze serveru. Publikování, tokeny doručení a ztracené události připojení se doručují do metod 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" na příjmu publikací. Stav relace také zahrnuje odběry vytvořené klientem MQTT. Můžete zvolit spuštění klienta MQTT s nebo bez udržování informací o stavu mezi relacemi. Změňte režim čisté relace nastavením `MqttConnectOptions.cleanSession` před připojením.

Identifikátor klienta

Tokeny doručení

Datum poslední vůle a potvrzení
Pokud došlo k neočekávanému ukončení připojení klienta MQTT , můžete produkt WebSphere MQ Telemetry nakonfigurovat tak, aby odeslal publikaci "last will and testament". Předefinujte obsah publikace a téma, kterému jej chcete odeslat. "Poslední vůle a testament" je vlastnost připojení. Vytvořte jej před připojením klienta.

Publikace
Publikace jsou instance `MqttMessage` , které jsou přidruženy k řetězci tématu. Klient produktu MQTT může vytvářet publikace k odesílání do produktu IBM WebSphere MQa k odběru témat v produktu IBM WebSphere MQ MQ za přihlášení k odběru publikací.

Kvality služby poskytované klientem MQTT
Klient produktu MQTT poskytuje tři úrovně kvality služby pro dodávání publikací do produktu WebSphere MQ a do klienta MQTT : "maximálně jednou", "alespoň jednou" a "přesně jednou". Pokud klient produktu MQTT odešle požadavek na produkt WebSphere MQ pro vytvoření odběru, odešle se požadavek s kvalitou služby "alespoň jednou".

Zachovaná publikování a klienti MQTT
Pokud vytvoříte odběr tématu, které má zachované publikování, bude vám okamžitě předáno nejnovější zachované publikování v tématu.

Odběry
Vytvoření odběrů pro registraci zájmu v tématech publikování pomocí filtru témat. Klient může vytvořit více odběrů nebo odběr obsahující filtr témat, který používá zástupné znaky, a zaregistrovat tak zájem o více témat. Publikace na témata, která odpovídají filtrům, se posílají na klienta. Odběry mohou zůstat aktivní, když je klient odpojen. Publikace se odesílají klientovi, jakmile se znovu připojí.

Řetězce témat a filtry témat v klientech MQTT
Řetězce témat a filtry témat se používají pro publikování a odběr. Syntaxe řetězců témat a filtrů v klientech produktu MQTT je z velké části stejné jako řetězce témat v produktu IBM WebSphere MQ.

Publikace

Publikace jsou instance `MqttMessage` , které jsou přidruženy k řetězci tématu. Klient produktu MQTT může vytvářet publikace k odesílání do produktu IBM WebSphere MQa k odběru témat v produktu IBM WebSphere MQ MQ za přihlášení k odběru publikací.

MqttMessage má jako svůj informační obsah pole bajtů. Zamířit, aby zprávy byly co nejmenší. Maximální délka zprávy povolené protokolem MQTT je 250 MB.

Klientský program MQTT obvykle používá java.lang.String nebo java.lang.StringBuffer k manipulaci s obsahem zpráv. Pro usnadnění práce má třída MqttMessage metodu toString pro převod jejího informačního obsahu na řetězec. Chcete-li vytvořit informační obsah bajtového pole z java.lang.String nebo java.lang.StringBuffer, použijte metodu getBytes .

Metoda getBytes převádí řetězec na výchozí znakovou sadu pro platformu. Standardní znaková sada je obecně UTF-8. Příručky MQTT , které obsahují pouze text, jsou obvykle kódovány v produktu UTF-8. Chcete-li potlačit výchozí znakovou sadu, použijte metodu getBytes("UTF8") .

V produktu IBM WebSphere MQje publikace MQTT přijata jako zpráva jms-bytes . Zpráva obsahuje složku MQRFH2 obsahující složku <mqtt>a složku <mqps> . Složka <mqtt> obsahuje položku clientId a qos, ale tento obsah se může v budoucnu měnit.

MqttMessage má tři další atributy: kvalitu služby, ať už je uchován, a zda je duplikát. Duplicitní příznak je nastaven pouze v případě, že kvalita služby je "alespoň jednou", nebo "přesně jednou". Pokud byla zpráva poslána dříve a klientem MQTT nebyla dostatečně rychle potvrzena, zpráva se odešle znovu s duplicitním atributem nastaveným na true.

Publikování

Chcete-li vytvořit publikování v aplikaci klienta MQTT , vytvořte MqttMessage. Nastavte informační obsah, kvalitu služby a informace o tom, zda je uchován, a volejte metodu MqttTopic.publish(MqttMessage message) , je vrácena MqttDeliveryToken a dokončení publikování je asynchronní.

Případně může klient produktu MQTT vytvořit dočasný objekt zprávy z parametrů v metodě MqttTopic.publish(byte [] payload, int qos, boolean retained) při vytváření publikace.

Pokud má publikování alespoň jednu úroveň kvality služeb QoS=1 nebo QoS=2nebo "právě jednou", zavolá klient MQTT rozhraní MqttClientPersistence . Zavolá produkt MqttClientPersistence , aby uložil zprávu před vrácením doručovacího tokenu do aplikace.

Aplikace se může rozhodnout blokovat, dokud nebude zpráva doručena na server, pomocí metody MqttDeliveryToken.waitForCompletion . Alternativně může aplikace pokračovat bez blokování. Chcete-li zkontrolovat, zda jsou publikace doručeny, bez blokování, registrujte instanci třídy zpětného volání, která implementuje MqttCallback s klientem MQTT . Klient MQTT volá metodu MqttCallback.deliveryComplete , jakmile je publikace doručena. V závislosti na kvalitě služby může být doručení téměř okamžité pro QoS=0, nebo může nějakou dobu trvat, než QoS=2.

Pokud je doručení dokončeno, použijte metodu MqttDeliveryToken.isComplete . Zatímco hodnota MqttDeliveryToken.isComplete je false, můžete volat MqttDeliveryToken.getMessage pro získání obsahu zprávy. Pokud je výsledek volání MqttDeliveryToken.isComplete true, zpráva byla vyřazena a zavoláním příkazu MqttDeliveryToken.getMessage by došlo k výjimce ukazatele null. Mezi MqttDeliveryToken.getMessage a MqttDeliveryToken.isCompleteneexistuje žádná vestavěná synchronizace.

Pokud se klient odpojí před přijetím všech nevyřízených tokenů doručení, může se před připojením dotazovat nová instance klienta na nevyřízené tokeny doručení. Dokud se klient nepřipojí, nejsou dokončeny žádné nové dodávky a je bezpečné volat MqttDeliveryToken.getMessage. Chcete-li zjistit, které publikace nebyly doručeny, použijte metodu MqttDeliveryToken.getMessage . Nevyřízené tokeny doručení jsou zrušeny, pokud se připojíte k výchozí hodnotě MqttConnectOptions.cleanSession , true.

přihlášení odběru

Správce front nebo produkt IBM MessageSight je odpovědný za vytváření publikování pro odeslání na odběratele produktu MQTT . Správce front kontroluje, zda filtr témat v odběru vytvořeném klientem MQTT odpovídá řetězci tématu v publikování. Shoda může být buď přesná shoda, nebo může shoda obsahovat zástupné znaky. Před postoupením publikace odběrateli správce front zkontroluje správce front atributy

tématu přidružené k této publikaci. Následuje postup vyhledávání popsaný v tématu Přihlášení k odběru pomocí řetězce tématu, který obsahuje zástupné znaky k identifikaci, zda objekt administrativního tématu uděluje oprávnění uživatele k odběru.

Když klient MQTT obdrží publikování s "alespoň jednou" kvalitou služby, volá metodu `MqttCallback.messageArrived` , aby zpracují publikování. Je-li kvalita služby publikace "přesně jednou", QoS=2, klient MQTT volá rozhraní produktu `MqttClientPersistence` , aby uložil zprávu při jejím přijetí. Pak zavolá příkaz `MqttCallback.messageArrived`.

Související pojmy

Zpětná volání a synchronizace v klientských aplikacích MQTT
Programovací model klienta produktu MQTT rozsáhle používá podprocesy. Podprocesy oddělující aplikaci klienta MQTT , stejně jako mohou, z prodlev při přenosu zpráv ze serveru a ze serveru. Publikování, tokeny doručení a ztracené události připojení se doručují do metod 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" na příjmu publikací. Stav relace také zahrnuje odběry vytvořené klientem MQTT. Můžete zvolit spuštění klienta MQTT s nebo bez udržování informací o stavu mezi relacemi. Změňte režim čisté relace nastavením `MqttConnectOptions.cleanSession` před připojením.

Identifikátor klienta

Tokeny doručení

Datum poslední vůle a potvrzení
Pokud došlo k neočekávanému ukončení připojení klienta MQTT , můžete produkt WebSphere MQ Telemetry nakonfigurovat tak, aby odeslal publikaci "last will and testament". Předefinujte obsah publikace a téma, kterému jej chcete odeslat. "Poslední vůle a testament" je vlastnost připojení. Vytvořte jej před připojením klienta.

Perzistence zpráv v klientech produktu MQTT

Kvality služby poskytované klientem MQTT
Klient produktu MQTT poskytuje tři úrovně kvality služby pro dodávání publikací do produktu WebSphere MQ a do klienta MQTT : "maximálně jednou", "alespoň jednou" a "přesně jednou". Pokud klient produktu MQTT odešle požadavek na produkt WebSphere MQ pro vytvoření odběru, odešle se požadavek s kvalitou služby "alespoň jednou".

Zachovaná publikování a klienti MQTT
Pokud vytvoříte odběr tématu, které má zachované publikování, bude vám okamžitě předáno nejnovější zachované publikování v tématu.

Odběry
Vytvoření odběrů pro registraci zájmu v tématech publikování pomocí filtru témat. Klient může vytvořit více odběrů nebo odběr obsahující filtr témat, který používá zástupné znaky, a zaregistrovat tak zájem o více témat. Publikace na témata, která odpovídají filtrům, se posílají na klienta. Odběry mohou zůstat aktivní, když je klient odpojen. Publikace se odesílají klientovi, jakmile se znovu připojí.

Řetězce témat a filtry témat v klientech MQTT
Řetězce témat a filtry témat se používají pro publikování a odběr. Syntaxe řetězců témat a filtrů v klientech produktu MQTT je z velké části stejné jako řetězce témat v produktu IBM WebSphere MQ.

Kvality služby poskytované klientem MQTT

Klient produktu MQTT poskytuje tři úrovně kvality služby pro dodávání publikací do produktu WebSphere MQ a do klienta MQTT : "maximálně jednou", "alespoň jednou" a "přesně jednou". Pokud klient produktu MQTT odešle požadavek na produkt WebSphere MQ pro vytvoření odběru, odešle se požadavek s kvalitou služby "alespoň jednou".

Quality of service of a publication is an attribute of `MqttMessage`. Je nastaven pomocí metody `MqttMessage.setQos`.

Metoda `MqttClient.subscribe` může snížit kvalitu služby aplikovanou na publikace odeslané klientovi na téma. Kvalita služeb publikace postoupené odběrateli se může lišit od kvality služby publikace. Nižší z těchto dvou hodnot se používá k předání publikace.

Nejvíce jednou
`QoS=0`

Zpráva je doručena nejvíce jednou nebo není doručena vůbec. Její doručení po síti není potvrzeno.
Zpráva není uložena. Při odpojení klienta nebo selhání serveru může dojít ke ztrátě zprávy.
`QoS=0` je nejrychlejší způsob přenosu. Někdy se říká "oheň a zapomenout".
Protokol MQTT nevyžaduje od serverů předávání publikačních publikací na straně `QoS=0` klientovi. Pokud je klient odpojen v době, kdy server obdrží publikování, může být publikování zrušeno, v závislosti na serveru. Služba telemetrie (MQXR) nevyřazovat zprávy odeslané s produktem `QoS=0`. Jsou uloženy jako přechodné zprávy a jsou zahozeny pouze v případě, že je správce front zastaven.

Nejméně jednou
`QoS=1`

`QoS=1` je výchozí režim přenosu.
Zpráva se vždy doručí alespoň jednou. Pokud odesilatel neobdrží potvrzení, je zpráva odeslána znovu s příznakem `DUP` , dokud není přijato potvrzení. Jako příjemce lze vícekrát odeslat stejnou zprávu a může ji několikrát zpracovat.
Zpráva musí být uložena lokálně na odesílateli a přijímači, dokud nebude zpracována.
Zpráva se odstraní z přijímače poté, co zpracoval zprávu. Je-li příjemcem zprostředkovatel, zpráva se publikuje na své odběratele. Je-li příjemcem klient, zpráva se doručí do aplikace odběratele. Jakmile je zpráva odstraněna, příjemce odešle potvrzení odesilateli.
Zpráva se odstraní od odesílatele poté, co mu bylo přijato potvrzení od příjemce.

Přesně jednou
`QoS=2`

Zpráva se vždy doručí přesně jednou.
Zpráva musí být uložena lokálně na odesílateli a přijímači, dokud nebude zpracována.
`QoS=2` je nejbezpečnější, ale nejpomalejší způsob přenosu. Mezi odesilatelem a příjemcem je třeba provést alespoň dva páry přenosů před tím, než je zpráva odstraněna od odesílatele. Zprávu lze na příjemce zpracovat po prvním přenosu.
V první dvojici přenosů odešle odesilatel zprávu a získá potvrzení od příjemce, že uložil zprávu. Pokud odesilatel neobdrží potvrzení, je zpráva odeslána znovu s příznakem `DUP` , dokud není přijato potvrzení.
Ve druhém páru přenosů odesilatel sděluje přijímači, že může dokončit zpracování zprávy, `"PUBREL"`. Pokud odesilatel neobdrží potvrzení o zprávě `"PUBREL"` , je zpráva `"PUBREL"` poslána znovu, dokud neobdrží potvrzení. Odesilatel odstraní zprávu, která byla uložena, když přijme potvrzení pro zprávu `"PUBREL"` .
Příjemce může zprávu zpracovat v první nebo druhé fázi za předpokladu, že tuto zprávu nezpracuje znovu. Je-li příjemcem zprostředkovatel, publikuje zprávu pro odběratele. Je-li příjemcem klient, doručí zprávu do aplikace odběratele. Příjemce pošle zprávu o dokončení zpět odesílateli, že dokončil zpracování zprávy.

Související pojmy

Zpětná volání a synchronizace v klientských aplikacích MQTT
Programovací model klienta produktu MQTT rozsáhle používá podprocesy. Podprocesy oddělující aplikaci klienta MQTT , stejně jako mohou, z prodlev při přenosu zpráv ze serveru a ze serveru. Publikování, tokeny doručení a ztracené události připojení se doručují do metod 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" na příjmu publikací. Stav relace také zahrnuje odběry vytvořené klientem MQTT. Můžete zvolit spuštění klienta

MQTT s nebo bez udržování informací o stavu mezi relacemi. Změňte režim čisté relace nastavením `MqttConnectOptions.cleanSession` před připojením.

Identifikátor klienta

Tokeny doručení

Datum poslední vůle a potvrzení
Pokud došlo k neočekávanému ukončení připojení klienta MQTT , můžete produkt WebSphere MQ Telemetry nakonfigurovat tak, aby odeslal publikaci "last will and testament". Předefinujte obsah publikace a téma, kterému jej chcete odeslat. "Poslední vůle a testament" je vlastnost připojení. Vytvořte jej před připojením klienta.

Perzistence zpráv v klientech produktu MQTT

Publikace
Publikace jsou instance `MqttMessage` , které jsou přidruženy k řetězci tématu. Klient produktu MQTT může vytvářet publikace k odesílání do produktu IBM WebSphere MQa k odběru témat v produktu IBM WebSphere MQ MQ za přihlášení k odběru publikací.

Zachovaná publikování a klienti MQTT
Pokud vytvoříte odběr tématu, které má zachované publikování, bude vám okamžitě předáno nejnovější zachované publikování v tématu.

Odběry
Vytvoření odběrů pro registraci zájmu v tématech publikování pomocí filtru témat. Klient může vytvořit více odběrů nebo odběr obsahující filtr témat, který používá zástupné znaky, a zaregistrovat tak zájem o více témat. Publikace na témata, která odpovídají filtrům, se posílají na klienta. Odběry mohou zůstat aktivní, když je klient odpojen. Publikace se odesílají klientovi, jakmile se znovu připojí.

Řetězce témat a filtry témat v klientech MQTT
Řetězce témat a filtry témat se používají pro publikování a odběr. Syntaxe řetězců témat a filtrů v klientech produktu MQTT je z velké části stejné jako řetězce témat v produktu IBM WebSphere MQ.

Zachovaná publikování a klienti MQTT

Pokud vytvoříte odběr tématu, které má zachované publikování, bude vám okamžitě předáno nejnovější zachované publikování v tématu.

Pomocí metody `MqttMessage.setRetained` lze určit, zda má být publikování na téma zachováno, či nikoli.

Chcete-li odstranit zachované publikování v produktu IBM WebSphere MQ, spusťte příkaz MQSC **CLEAR TOPICSTR** .

Pokud vytvoříte publikování s informačním obsahem s hodnotou null, bude odběratelům předáno prázdné publikování. Ostatní zprostředkovatelé produktu MQTT mohou nepředat prázdnou publikaci odběratelům.

Pokud publikujete nezachované publikování na téma, které má zachované publikování, zachované publikování nebude mít vliv na zachované publikování. Aktuální odběratelé obdrží novou publikaci. Noví odběratelé obdrží zachované publikování jako první a poté obdrží nové publikace.

Když vytvoříte nebo aktualizujete zachované publikování, odešlete publikaci se QoS nebo 1 nebo 2. Pokud ji odešlete pomocí QoS z 0, produkt IBM WebSphere MQ vytvoří netrvalé zachované publikování. Publikování nebude zachováno, je-li správce front zastaven.

Použijte zachovaná publikování a zaznamenejte nejnovější hodnotu měření. Noví odběratelé uchovaného tématu okamžitě obdrží nejnovější hodnotu měření. Pokud od odběratele, který byl naposledy přihlášen k tématu publikování, nebyla provedena žádná nová měření, a pokud se odběratel znovu přihlásí, odběratel obdrží nejnovější zachované publikování v rámci tématu znovu.

Související pojmy

Zpětná volání a synchronizace v klientských aplikacích MQTT
Programovací model klienta produktu MQTT rozsáhle používá podprocesy. Podprocesy oddělující aplikaci klienta MQTT , stejně jako mohou, z prodlev při přenosu zpráv ze serveru a ze serveru. Publikování, tokeny doručení a ztracené události připojení se doručují do metod 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" na příjmu publikací. Stav relace také zahrnuje odběry vytvořené klientem MQTT. Můžete zvolit spuštění klienta MQTT s nebo bez udržování informací o stavu mezi relacemi. Změňte režim čisté relace nastavením `MqttConnectOptions.cleanSession` před připojením.

Identifikátor klienta

Tokeny doručení

Datum poslední vůle a potvrzení
Pokud došlo k neočekávanému ukončení připojení klienta MQTT , můžete produkt WebSphere MQ Telemetry nakonfigurovat tak, aby odeslal publikaci "last will and testament". Předefinujte obsah publikace a téma, kterému jej chcete odeslat. "Poslední vůle a testament" je vlastnost připojení. Vytvořte jej před připojením klienta.

Perzistence zpráv v klientech produktu MQTT

Publikace
Publikace jsou instance `MqttMessage` , které jsou přidruženy k řetězci tématu. Klient produktu MQTT může vytvářet publikace k odesílání do produktu IBM WebSphere MQa k odběru témat v produktu IBM WebSphere MQ MQ za přihlášení k odběru publikací.

Kvality služby poskytované klientem MQTT
Klient produktu MQTT poskytuje tři úrovně kvality služby pro dodávání publikací do produktu WebSphere MQ a do klienta MQTT : "maximálně jednou", "alespoň jednou" a "přesně jednou". Pokud klient produktu MQTT odešle požadavek na produkt WebSphere MQ pro vytvoření odběru, odešle se požadavek s kvalitou služby "alespoň jednou".

Odběry
Vytvoření odběrů pro registraci zájmu v tématech publikování pomocí filtru témat. Klient může vytvořit více odběrů nebo odběr obsahující filtr témat, který používá zástupné znaky, a zaregistrovat tak zájem o více témat. Publikace na témata, která odpovídají filtrům, se posílají na klienta. Odběry mohou zůstat aktivní, když je klient odpojen. Publikace se odesílají klientovi, jakmile se znovu připojí.

Řetězce témat a filtry témat v klientech MQTT
Řetězce témat a filtry témat se používají pro publikování a odběr. Syntaxe řetězců témat a filtrů v klientech produktu MQTT je z velké části stejné jako řetězce témat v produktu IBM WebSphere MQ.

Odběry

Vytvoření odběrů pro registraci zájmu v tématech publikování pomocí filtru témat. Klient může vytvořit více odběrů nebo odběr obsahující filtr témat, který používá zástupné znaky, a zaregistrovat tak zájem o více témat. Publikace na témata, která odpovídají filtrům, se posílají na klienta. Odběry mohou zůstat aktivní, když je klient odpojen. Publikace se odesílají klientovi, jakmile se znovu připojí.

Vytvořte odběry pomocí metod `MqttClient.subscribe` , které jsou předáním jednoho nebo více filtrů témat a parametrů kvality služby. Parametr quality of service nastavuje maximální kvalitu služby, kterou je odběratel připraven použít pro příjem zprávy. Zprávy odeslané tomuto klientovi nemohou být doručeny s vyšší kvalitou služby. Kvalita služby je nastavena na nižší z původní hodnoty, když byla zpráva publikována a úroveň uvedená pro odběr. Výchozí kvalita služby pro příjem zpráv je `QoS=1`, alespoň jednou.

Samotný požadavek na odběr je odeslán s produktem `QoS=1`.

Publications přijímá odběratel, když klient MQTT volá metodu `MqttCallback.messageArrived` . Metoda `messageArrived` také předává řetězec tématu, se kterým byla zpráva publikována, na odběratele.

Pomocí metod `MqttClient.unsubscribe` můžete odebrat odběr nebo sadu či odběry.

Příkaz WebSphere MQ může odebrat odběr. Seznam odběrů pomocí programu Průzkumník produktu WebSphere MQ nebo pomocí příkazů **runmqsc** nebo PCF. Všechny odběry klienta MQTT mají název. Zobrazí se název formuláře: *`ClientIdentifier:Topic name`*

Používáte-li výchozí hodnotu `MqttConnectOptions` nebo před připojením klienta nastavíte `MqttConnectOptions.cleanSession` na hodnotu `true`, budou všechny staré odběry klienta při připojení odebrány. Všechny nové odběry, které klient během relace vytvoří, jsou při odpojení odebrány.

Pokud před připojením nastavíte `MqttConnectOptions.cleanSession` na hodnotu `false`, budou všechny vytvořené odběry klienta přidány ke všem odběrům, které existovaly pro klienta 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 je třeba nastavit režim `cleanSession`; režim trvá pro celou relaci. Chcete-li změnit jeho nastavení, je třeba klienta odpojit a znovu připojit. Pokud změníte režimy z používání `cleanSession=false` na `cleanSession=true`, budou všechny předchozí odběry pro klienta a všechny publikace, které nebyly přijaty, vyřazeny.

Publikace, které odpovídají aktivním odběrům, se odešlou klientovi, jakmile jsou publikovány. Je-li klient odpojen, odešle se klientovi, pokud se znovu připojí ke stejnému serveru se stejným identifikátorem klienta a `MqttConnectOptions.cleanSession` nastaveným na `false`.

Odběry pro určitého klienta jsou identifikovány identifikátorem klienta. Klienta můžete znovu připojit z jiného klientského zařízení na stejný server a pokračovat se stejnými odběry a přijímat nedoručené publikace.

Související pojmy

Zpětná volání a synchronizace v klientských aplikacích MQTT
Programovací model klienta produktu MQTT rozsáhle používá podprocesy. Podprocesy oddělující aplikaci klienta MQTT , stejně jako mohou, z prodlev při přenosu zpráv ze serveru a ze serveru. Publikování, tokeny doručení a ztracené události připojení se doručují do metod 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" na příjmu publikací. Stav relace také zahrnuje odběry vytvořené klientem MQTT. Můžete zvolit spuštění klienta MQTT s nebo bez udržování informací o stavu mezi relacemi. Změňte režim čisté relace nastavením `MqttConnectOptions.cleanSession` před připojením.

Identifikátor klienta

Tokeny doručení

Datum poslední vůle a potvrzení
Pokud došlo k neočekávanému ukončení připojení klienta MQTT , můžete produkt WebSphere MQ Telemetry nakonfigurovat tak, aby odeslal publikaci "last will and testament". Předefinujte obsah publikace a téma, kterému jej chcete odeslat. "Poslední vůle a testament" je vlastnost připojení. Vytvořte jej před připojením klienta.

Perzistence zpráv v klientech produktu MQTT

Publikace
Publikace jsou instance `MqttMessage` , které jsou přidruženy k řetězci tématu. Klient produktu MQTT může vytvářet publikace k odesílání do produktu IBM WebSphere MQa k odběru témat v produktu IBM WebSphere MQ MQ za přihlášení k odběru publikací.

Kvality služby poskytované klientem MQTT
Klient produktu MQTT poskytuje tři úrovně kvality služby pro dodávání publikací do produktu WebSphere MQ a do klienta MQTT : "maximálně jednou", "alespoň jednou" a "přesně jednou". Pokud klient produktu MQTT odešle požadavek na produkt WebSphere MQ pro vytvoření odběru, odešle se požadavek s kvalitou služby "alespoň jednou".

Zachovaná publikování a klienti MQTT

Pokud vytvoříte odběr tématu, které má zachované publikování, bude vám okamžitě předáno nejnovější zachované publikování v tématu.

Řetězce témat a filtry témat v klientech MQTT
Řetězce témat a filtry témat se používají pro publikování a odběr. Syntaxe řetězců témat a filtrů v klientech produktu MQTT je z velké části stejné jako řetězce témat v produktu IBM WebSphere MQ.

Řetězce témat a filtry témat v klientech MQTT

Řetězce témat a filtry témat se používají pro publikování a odběr. Syntaxe řetězců témat a filtrů v klientech produktu MQTT je z velké části stejné jako řetězce témat v produktu IBM WebSphere MQ.

Řetězce témat se používají k odeslání publikování na odběratele. Vytvořte řetězec tématu za použití metody `MqttClient.getTopic(java.lang.String topicString)`.

Filtry témat se používají k odběru témat a přijímat publikování. Filtry témat mohou obsahovat zástupné znaky. Se zástupnými znaky se můžete přihlásit k odběru více témat. Vytvořte filtr témat pomocí metody odběru; například `MqttClient.subscribe(java.lang.String topicFilter)`.

Řetězce tématu

Syntaxe řetězce tématu IBM WebSphere MQ je popsána v tématu Řetězce tématu. Syntaxe řetězců témat MQTT je popsána ve třídě `MqttClient` v dokumentaci k rozhraním API pro Klient MQTT pro produkt Java. Pro odkazy na dokumentaci ke klientským rozhraním API pro knihovny klienta MQTT viz Odkaz na programování klienta MQTT.

Syntaxe jednotlivých typů řetězců témat je téměř identická. Jsou zde čtyři menší rozdíly:

1. Topic strings sent to IBM WebSphere MQ by MQTT clients must follow the convention for queue manager names. Zejména řetězce témat nemohou obsahovat pomlčky.
2. Maximální délka se liší. Řetězce témat IBM WebSphere MQ jsou omezeny na 10,240 znaků. Klient MQTT může vytvořit řetězce témat až 65535 bajtů.
3. Řetězec tématu vytvořený klientem MQTT nemůže obsahovat znak null.
4. V produktu WebSphere Message Broker byla úroveň tématu null, `'...//...'` byla neplatná. Hodnoty témat s hodnotou null jsou podporovány produktem IBM WebSphere MQ.

Na rozdíl od IBM WebSphere MQ publish/subscribe, `mqttv3` protokol nemá koncepci objektu administrativního tématu. Řetězec tématu nelze zkonstruovat z objektu tématu a z řetězce tématu. Řetězec tématu je však mapován na administrativní téma v produktu IBM WebSphere MQ. Řízení přístupu přidružené k administrativnímu tématu určuje, zda je publikování publikováno do tématu, nebo zrušeno. Atributy, které jsou použity ke zveřejnění při jeho předání odběratelům, jsou ovlivněny atributy administrativního tématu.

Filtry témat

Syntaxe filtru témat IBM WebSphere MQ je popsána v tématu Schéma zástupných znaků na základě témat. Syntaxe filtrů témat, které lze sestavit s klientem MQTT , je popsána ve třídě `MqttClient` v dokumentaci k rozhraním API pro Klient MQTT pro produkt Java. Pro odkazy na dokumentaci ke klientským rozhraním API pro knihovny klienta MQTT viz Odkaz na programování klienta MQTT.

Syntaxe jednotlivých typů filtrů témat je téměř identická. Jediný rozdíl je v tom, jak různí zprostředkovatelé MQTT interpretují filtr témat. V produktu WebSphere Message Broker V6lze zástupný znak s více úrovněmi použít pouze na konci filtru témat. V produktu IBM WebSphere MQlze zástupný znak více úrovní použít na libovolné úrovni stromu témat, například `USA/#/Dutchess County`.

Související pojmy

Zpětná volání a synchronizace v klientských aplikacích MQTT
Programovací model klienta produktu MQTT rozsáhle používá podprocesy. Podprocesy oddělující aplikaci klienta MQTT , stejně jako mohou, z prodlev při přenosu zpráv ze serveru a ze serveru. Publikování, tokeny doručení a ztracené události připojení se doručují do metod 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" na příjmu publikací. Stav relace také zahrnuje odběry vytvořené klientem MQTT. Můžete zvolit spuštění klienta MQTT s nebo bez udržování informací o stavu mezi relacemi. Změňte režim čisté relace nastavením `MqttConnectOptions.cleanSession` před připojením.

Identifikátor klienta

Tokeny doručení

Datum poslední vůle a potvrzení
Pokud došlo k neočekávanému ukončení připojení klienta MQTT , můžete produkt WebSphere MQ Telemetry nakonfigurovat tak, aby odeslal publikaci "last will and testament". Předefinujte obsah publikace a téma, kterému jej chcete odeslat. "Poslední vůle a testament" je vlastnost připojení. Vytvořte jej před připojením klienta.

Perzistence zpráv v klientech produktu MQTT

Publikace
Publikace jsou instance `MqttMessage` , které jsou přidruženy k řetězci tématu. Klient produktu MQTT může vytvářet publikace k odesílání do produktu IBM WebSphere MQa k odběru témat v produktu IBM WebSphere MQ MQ za přihlášení k odběru publikací.

Kvality služby poskytované klientem MQTT
Klient produktu MQTT poskytuje tři úrovně kvality služby pro dodávání publikací do produktu WebSphere MQ a do klienta MQTT : "maximálně jednou", "alespoň jednou" a "přesně jednou". Pokud klient produktu MQTT odešle požadavek na produkt WebSphere MQ pro vytvoření odběru, odešle se požadavek s kvalitou služby "alespoň jednou".

Zachovaná publikování a klienti MQTT
Pokud vytvoříte odběr tématu, které má zachované publikování, bude vám okamžitě předáno nejnovější zachované publikování v tématu.

Odběry
Vytvoření odběrů pro registraci zájmu v tématech publikování pomocí filtru témat. Klient může vytvořit více odběrů nebo odběr obsahující filtr témat, který používá zástupné znaky, a zaregistrovat tak zájem o více témat. Publikace na témata, která odpovídají filtrům, se posílají na klienta. Odběry mohou zůstat aktivní, když je klient odpojen. Publikace se odesílají klientovi, jakmile se znovu připojí.

Koncepty programování klienta C

V tomto tématu jsou popsány rozdíly mezi klientem jazyka C a prostředím Java pro verzi 3.1 přenosu MQ Telemetry. Téma doplňuje koncepci klienta a referenční informace C.

Téma je organizováno stejným způsobem jako “Koncepty programování klienta MQTT” na stránce 507. Každá hlavička odpovídá tématu *WebSphere(r) MQ Telemetry Transport Programming concepts*. Sekce popisují rozdíly mezi klientem jazyka C a klientem Java. Rozdíly mezi metodami jazyka Java a funkcemi jazyka C nejsou popsány v malých rozdílech.

Klient jazyka C se nejčastěji používá k implementaci odlehčeného adaptéru mezi telemetrickým zařízením a démonem WebSphere MQ Telemetry pro zařízení. Démon se běžně používá jako koncentrátor sítě mezi velmi lehkými telemetrií a službami telemetrie (MQXR).

Démon WebSphere MQ Telemetry pro zařízení je také klientem jazyka C a jsou popsány rozdíly v chování této služby telemetrií (MQXR). Démon neposkytuje implementaci JAAS nebo zabezpečení SSL pro klienty, kteří se k němu připojují.

`mqttclient.dll` a `mqttclient.lib` jsou 32bitové knihovny systému Windows, které obsahují funkce klienta pro implementaci jazyka C MQ Telemetry Transport verze 3.1 . 32bitovou knihovnu Linux jsou `libmqttclient.so` a `libmqttclient.a`. Dva hlavičkové soubory obsahují funkci a další deklarace potřebné pro klientské aplikace: `MQTTClient.h` a `MQTTClientPersistence.h`. Tyto soubory se poskytují spolu s instalací produktu WebSphere MQ Telemetry.

Chcete-li vyvinout a spustit klienta přenosu MQ Telemetry, je třeba zkopírovat tyto soubory na klientské zařízení. Na rozdíl od klientů WebSphere MQ není třeba instalovat oddělené běhové prostředí klienta.

Konzultujte podmínky licencování přidružené k funkci WebSphere MQ Telemetry , která řídí připojení transportních klientů MQ Telemetry k produktu WebSphere MQ a démonu WebSphere MQ Telemetry pro zařízení.

Klient jazyka C je referenční implementace verze 3.1 přenosu MQ Telemetry. Můžete implementovat vlastní klienty v různých jazycích vhodných pro různé platformy zařízení. Podrobnosti naleznete v dokumentu MQ Telemetry Transport format and protocol .

Identifikátor klienta MQTT

“Identifikátor klienta” na stránce 512	Identifikátor klienta je 23bajtový řetězec, který identifikuje klienta MQTT . Každý identifikátor musí být jedinečný, tedy platný ve stejnou dobu pouze pro jediného připojeného klienta. Identifikátor musí obsahovat pouze znaky platné v názvu správce front. V rámci těchto omezení budete moci používat libovolný identifikační řetězec. Je důležité mít proceduru pro přidělování identifikátorů klienta a prostředků ke konfiguraci klienta se zvoleným identifikátorem.

- Žádné rozdíly.

Publikace

“Publikace” na stránce 518	Publikace jsou instance `MqttMessage` , které jsou přidruženy k řetězci tématu. Klient produktu MQTT

- Funkce zpětného volání není volána pro publikování s publikacemi "fire and forget", `QoS=0`, quality of service.

Tokeny doručení

“Tokeny doručení” na stránce 513	Když klient publikuje na téma nový token doručení, dojde k vytvoření nového tokenu doručení. Použijte token doručení k monitorování doručení publikování nebo k blokování klientské aplikace, dokud nebude doručení dokončeno.

- Token doručení je `int`. Má typedef z `MQTTClient_deliveryToken`
- Funkce zpětného volání není volána pro publikování s publikacemi "fire and forget", `QoS=0`, quality of service.

Zachovaná publikování

“Zachovaná publikování a klienti MQTT” na stránce 522	Pokud vytvoříte odběr tématu, které má zachované publikování, bude vám okamžitě předáno nejnovější zachované publikování v tématu.

- Zachovaná zpráva se uloží pouze v démonu, je-li perzistence konfigurována; viz Ukládání zachovaných zpráv a odběrů.

  V případě produktu WebSphere MQovlivňuje kvalita služby, zda je uchovaná zpráva trvale uložena. Je-li klient připojen ke službě telemetrie, zadrží zprávy s "ohněm a zapomenem", služba kvality služby `QoS=0` se zruší, pokud se správce front vypne.

Odběry

“Odběry” na stránce 523	Vytvoření odběrů pro registraci zájmu v tématech publikování pomocí filtru témat. Klient může vytvořit více odběrů nebo odběr obsahující filtr témat, který používá zástupné znaky, a zaregistrovat tak zájem o více témat. Publikace na témata, která odpovídají filtrům, se posílají na klienta. Odběry mohou zůstat aktivní, když je klient odpojen. Publikace se odesílají klientovi, jakmile se znovu připojí.

- Trvalé odběry se ukládají do démona pouze v případě, že je perzistence konfigurována. Další informace naleznete v tématu Ukládání zachovaných zpráv a odběrů.
- Publikování lze přijímat synchronně. Volejte funkci `MQTTClient_receive` .

Zpětná volání a synchronizace

“Zpětná volání a synchronizace v klientských aplikacích MQTT” na stránce 508	Programovací model klienta produktu MQTT rozsáhle používá podprocesy. Podprocesy oddělující aplikaci klienta MQTT , stejně jako mohou, z prodlev při přenosu zpráv ze serveru a ze serveru. Publikování, tokeny doručení a ztracené události připojení se doručují do metod ve třídě zpětného volání, která implementuje `MqttCallback`.

- Operace synchronizace v klientovi C je modální. Volání `MQTTClient_setCallback` převede klienta do asynchronního režimu.
- V synchronním režimu musí aplikační klient dobrovolně zajistit řízení, aby mohl klient MQTT zpracovávat potvrzení a vydávat příkazy ping protokolu MQTT za účelem udržení aktivity sítě. Výtěžek řídí voláním `MQTTClient_receive` nebo `MQTTClient_yield`.

Řetězce a filtry témat

“Řetězce témat a filtry témat v klientech MQTT” na stránce 525	Řetězce témat a filtry témat se používají pro publikování a odběr. Syntaxe řetězců témat a filtrů v klientech produktu MQTT je z velké části stejné jako řetězce témat v produktu IBM WebSphere MQ.

- Démon WebSphere MQ Telemetry pro zařízení zpracovává zástupný znak více úrovní # odlišně od produktu WebSphere MQ v7. /# musí být poslední dva znaky v řetězci filtru pro # , aby se choval jako zástupný znak. V produktu WebSphere MQ v7je produkt `../#/..` platným použitím zástupného znaku více úrovní. Démon WebSphere MQ Telemetry pro zařízení považuje zástupný znak s více úrovněmi za stejný jako produkt WebSphere MQ Broker v6.

Kvalita služby

“Kvality služby poskytované klientem MQTT” na stránce 520	Klient produktu MQTT poskytuje tři úrovně kvality služby pro dodávání publikací do produktu WebSphere MQ a do klienta MQTT : "maximálně jednou", "alespoň jednou" a "přesně jednou". Pokud klient produktu MQTT odešle požadavek na produkt WebSphere MQ pro vytvoření odběru, odešle se požadavek s kvalitou služby "alespoň jednou".

- Žádné rozdíly.

Trvalost zpráv

“Perzistence zpráv v klientech produktu MQTT” na stránce 516	Publikační zprávy jsou prováděny jako trvalé, pokud jsou odeslány s kvalitou služby alespoň jednou, nebo "přesně jednou". Na klientovi můžete implementovat vlastní mechanismus perzistence nebo použít výchozí mechanismus perzistence, který je dodáván s klientem. Perzistence funguje v obou směrech, pro publikování odeslané na klienta nebo z klienta.

- Vzhledem k odlišnostem vazeb jazyka nastavte mechanizmus perzistence zpráv v klientu C následujícím způsobem. Volejte klienta MQTT C s jednou ze tří voleb, které jsou nastaveny jako čtvrtý parametr na hodnotu `MQTTClient_create`:

  `MQTTCLIENT_PERSISTENCE_DEFAULT`
  Perzistence založená na souboru, jejíž podrobnosti jsou specifické pro platformu klienta.

  `MQTTCLIENT_PERSISTENCE_NONE`
  Data jsou uchovávané pouze v paměti a jsou ztracena při zastavení klienta. Démon WebSphere MQ Telemetry pro zařízení podporuje pouze tuto volbu.

  `MQTTCLIENT_PERSISTENCE_USER`
  Můžete vyvinout funkce pro implementaci vlastního mechanizmu perzistence. Předejte strukturu, `MQTTClient_persistence` obsahující ukazatele na vaše funkce ve volání `MQTTClient_create` . Podrobnosti naleznete v informacích o odkazech na klienta protokolu MQTT C.

Vyčistit relace

“Vyčistit relace” na stránce 511	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" na příjmu publikací. Stav relace také zahrnuje odběry vytvořené klientem MQTT. Můžete zvolit spuštění klienta MQTT s nebo bez udržování informací o stavu mezi relacemi. Změňte režim čisté relace nastavením `MqttConnectOptions.cleanSession` před připojením.

- Žádné rozdíly.

Poslední zpráva při selhání

“Datum poslední vůle a potvrzení” na stránce 515	Pokud došlo k neočekávanému ukončení připojení klienta MQTT , můžete produkt WebSphere MQ Telemetry nakonfigurovat tak, aby odeslal publikaci "last will and testament". Předefinujte obsah publikace a téma, kterému jej chcete odeslat. "Poslední vůle a testament" je vlastnost připojení. Vytvořte jej před připojením klienta.

- Žádné rozdíly.

Obsluha chyb programu

Tyto informace vysvětlují chyby související s voláními MQI MQI buď při volání, nebo při doručení její zprávy do konečného cíle.

Kdykoli je to možné, vrátí správce front veškeré chyby, jakmile je provedeno volání MQI. Jedná se o *lokálně zjištěné chyby*.

Při odesílání zpráv do vzdálené fronty nemusí být při volání MQI zjevné chyby. V takovém případě správce front, který identifikuje chyby, ohlásí odeslání jiné zprávy do původního programu. Jedná se o *vzdáleně určené chyby*.

Lokálně určené chyby

Informace o lokálně určených chybách, které zahrnují: selhání při volání MQI, přerušení systému a zprávy obsahující nesprávná data.

Tři nejčastější příčiny chyb, které správce front může hlásit okamžitě, jsou:

- Selhání volání MQI; například, protože je plná fronta
- Přerušení běhu některé části systému, na které závisí vaše aplikace; například správce front.
- Zprávy obsahující data, která nelze úspěšně zpracovat

Používáte-li asynchronní prostředek vložení, chyby se neohlašují okamžitě. Použijte volání MQSTAT k načtení informací o stavu pro předchozí asynchronní operace put.

Selhání volání MQI

Správce front může okamžitě nahlásit případné chyby v kódování volání MQI. To se provádí pomocí souboru předdefinovaných návratových kódů. Ty jsou rozděleny do kódů dokončení a kódů příčiny.

Chcete-li zobrazit, zda je volání úspěšné, správce front vrátí při dokončení volání *kód dokončení* . K dispozici jsou tři kódy dokončení označující úspěch, částečné dokončení a selhání volání. Správce front také vrátí *kód příčiny* , který indikuje důvod částečného dokončení nebo selhání volání.

Kódy dokončení a příčiny pro každé volání jsou vypsány s popisem daného volání v tématu Návratové kódy. Podrobnější informace, včetně nápadů pro nápravnou akci, najdete v tématu:

- Kódy příčin pro všechny ostatní platformy WebSphere MQ

Navrhněte své programy ke zpracování všech návratových kódů, které mohou nastat při každém hovoru.

Přerušení systému

Pokud správce front, k němuž je připojen, se musí zotavit ze selhání systému, může být vaše aplikace nevědomá žádné přerušení. Je však třeba navrhnout aplikaci tak, aby nedošlo ke ztrátě dat, dojde-li k takovému přerušení.

Metody, které můžete použít k zajištění konzistence dat, závisí na platformě, na které je správce front spuštěn:

Systémy UNIX, Linux a Windows
V těchto prostředích můžete provést volání MQPUT a MQGET obvyklým způsobem, je však nutné deklarovat synchronizační body pomocí volání MQCMIT a MQBACK (viz “Potvrzení a zálohování jednotek práce” na stránce 310). V prostředí CICS jsou příkazy MQCMIT a MQBACK zakázány, protože můžete provést volání MQPUT a MQGET v rámci jednotek práce, které jsou spravovány produktem CICS.

Použít trvalé zprávy pro provedení všech dat, které si nemůžete dovolit ztratit. Trvalé zprávy jsou obnoveny ve frontách, pokud se správce front musí zotavit ze selhání. Při použití produktu WebSphere MQ v systémech UNIX, Linuxa Windows dojde k selhání volání MQGET nebo MQPUT v rámci vaší aplikace při zaplnění všech souborů protokolu se zprávou MQRC_RESOURCE_PROBLÉM. Další informace o souborech protokolu v systémech AIX, HP-UX, Linux, Solaris a Windows naleznete v části Administrace.

Pokud je správce front v době spuštění aplikace zastaven operátorem, je obvykle použita volba uvedení do klidového stavu. Správce front přejde do klidového stavu, ve kterém mohou aplikace pokračovat v práci, ale musí být ukončeny co nejdříve. Malé a rychlé aplikace mohou pravděpodobně ignorovat stav uvedení do klidového stavu a pokračovat, dokud nebudou ukončeny normální. Delší běžící aplikace nebo ty, které čekají na příchod zpráv, by měly použít volbu *selže při uvedení do klidového stavu* , když používají volání MQOPEN, MQPUT, MQPUT1a MQGET. Tyto volby znamenají, že volání selže, když správce front přechází do klidového stavu, ale aplikace může mít stále čas k tomu, aby mohla být ukončena čistě, vydáním volání, která ignorují stav uvedení do klidového stavu. Takové aplikace by mohly také potvrdit nebo vrátit zpět změny, které provedli, a poté ukončit.

Je-li správce front přinucen zastavit (tj. ukončit bez uvedení do klidového stavu), aplikace obdrží při volání MQI příkaz MQRC_CONNECTION_BROKEN kódu příčiny. Ukončete aplikaci nebo případně na systémech UNIX, Linuxa Windows zadejte volání MQDISC.

Zprávy obsahující nesprávná data

Použijete-li jednotky práce ve vaší aplikaci, pokud program nemůže úspěšně zpracovat zprávu, kterou načítá z fronty, je volání MQGET vráceny.

Správce front udržuje počet (v poli *BackoutCount* v deskriptoru zpráv) počtu případů, kdy k tomu dojde. Udržuje tento počet v deskriptoru každé zprávy, která je ovlivněna. Tento počet může poskytovat cenné informace o efektivitě aplikace. Zprávy s počtem odvolání, které se zvyšují v průběhu času, jsou opakovaně odmítány; navrhněte svou aplikaci tak, aby analyzovala příčiny těchto zpráv a ošetla tyto zprávy odpovídajícím způsobem.

V systému WebSphere MQ pro systémy Windows, UNIX a Linux platí, že počet vrácení vždy přežije restartované správce front.

Použití zpráv sestav k určování problémů

Vzdálený správce front nemůže hlásit chyby, jako je například selhání při vložení zprávy do fronty při provedení volání MQI, ale může vám odeslat zprávu s hlášením o tom, jak zprávu zpracoval.

Ve vaší aplikaci můžete vytvářet zprávy sestav (MQPUT) a také vybrat volbu jejich přijetí (v takovém případě jsou odesílány buď jinou aplikací, nebo správcem front).

Vytváření zpráv sestav

Zprávy hlášení umožňují aplikaci sdělit jiné aplikaci, že se nedokáže vypořádat se zprávou, která byla odeslána.

Pole *Report* však musí být nejprve analyzováno, aby bylo možné určit, zda aplikace, která zprávu odeslala, má zájem o informace o případných problémech. Poté, co jste určili, že je vyžadována zpráva sestavy, musíte se rozhodnout:

- Zda chcete zahrnout celou původní zprávu, pouze prvních 100 bajtů dat, nebo žádné z původní zprávy.
- Co se má provést s původní zprávou. Můžete ji vyřadit nebo nechat zahodit do fronty nedoručených zpráv.
- Zda je třeba také použít obsah polí *MsgId* a *CorrelId* .

Použijte pole *Feedback* , abyste označili důvod generování zprávy sestavy. Umístěte zprávy hlášení do fronty pro odpověď aplikace. Další informace najdete v tématu Zpětná vazba .

Vyžádání a příjem (MQGET) zpráv sestavy

Když odešlete zprávu do jiné aplikace, nebudete informováni o žádných problémech, dokud nedokončíte pole *Report* , abyste označili zpětnou vazbu, kterou požadujete. Dostupné volby naleznete v tématu Struktura pole sestavy .

Správci front vždy vloží zprávy sestav do fronty pro odpovědi aplikace a doporučuje se, aby vaše vlastní aplikace byly stejné. Použijete-li službu zpráv sestavy, zadejte název své odpovědi do fronty v deskriptoru zprávy vaší zprávy; jinak se volání MQPUT nezdaří.

Vaše aplikace musí obsahovat procedury, které monitorují vaši odpověď do fronty a zpracovávají zprávy, které do ní přicházejí. Nezapomeňte, že zpráva sestavy může obsahovat veškerou původní zprávu, prvních 100 bajtů původní zprávy nebo žádnou z původní zprávy.

Správce front nastaví pole *Feedback* ve zprávě sestavy tak, aby indikovalo příčinu chyby; například cílová fronta neexistuje. Vaše programy by měly provést totéž.

Další informace o zprávách sestav viz “Hlášení zpráv” na stránce 11.

Vzdáleně určené chyby

Při odesílání zpráv do vzdálené fronty, a to i v případě, že lokální správce front zpracoval volání MQI bez nalezení chyby, mohou jiné faktory ovlivnit způsob zpracování zprávy vzdáleným správcem front.

Například, fronta, kterou cílíte, může být plná, nebo nemusí existovat. Pokud má být vaše zpráva obsluhována jinými intermediačními správci front v přenosové cestě k cílové frontě, může některá z těchto informací nalézt chybu.

Problémy při doručování zprávy

Pokud se volání MQPUT nezdaří, můžete se pokusit o vložení zprávy do fronty znovu, vrátit ji odesilateli nebo ji umístit do fronty nedoručených zpráv.

Každá volba má své výhody, ale možná se nebudete chtít znovu pokusit o vložení zprávy z důvodu, že došlo k selhání příkazu MQPUT, protože cílová fronta byla plná. V této instanci umožňuje vložení fronty do fronty nedoručených zpráv později ji doručit do správné cílové fronty.

Zopakovat doručení zprávy

Před tím, než bude zpráva vložena do fronty nedoručených zpráv, se vzdálený správce front pokusí znovu vložit zprávu do fronty, pokud byly pro kanál nastaveny atributy *MsgRetryCount* a *MsgRetryInterval* , nebo pokud existuje uživatelský program opakování, který má být použit (jehož název je držen v poli atributu kanálu *MsgRetryExitId*).

Je-li pole *MsgRetryExitId* prázdné, použijí se hodnoty v atributech *MsgRetryCount* a *MsgRetryInterval* .

Není-li pole *MsgRetryExitId* prázdné, spustí se výstupní program tohoto názvu. Další informace o použití vlastních ukončovacích programů najdete v tématu “Kanály-uživatelské programy pro kanály systému zpráv” na stránce 380.

Vrátit zprávu odesílateli

Můžete vrátit zprávu odesílateli tak, že požádáte o vygenerování zprávy sestavy, aby zahrnovala všechny původní zprávy.

Podrobnosti o volbách zpráv sestav viz “Hlášení zpráv” na stránce 11 .

Použití fronty nedoručených zpráv (nedoručená zpráva)

Pokud správce front nemůže doručit zprávu, pokusí se vložit zprávu do fronty nedoručených zpráv. Tato fronta by měla být definována při instalaci správce front.

Vaše programy mohou používat frontu nedoručených zpráv stejným způsobem, jakým jej správce front používá. Název fronty nedoručených zpráv je možné najít otevřením objektu správce front (pomocí volání MQOPEN) a zjišťování informací o atributu *DeadLetterQName* (pomocí volání MQINQ).

Když správce front vloží do této fronty zprávu, přidá do zprávy záhlaví zprávy, jejíž formát je popsán strukturou záhlaví nedoručených zpráv (MQDLH); viz MQDLH-Dead-letter header . Do tohoto záhlaví patří název cílové fronty a důvod umístění zprávy do fronty nedoručených zpráv. Musí být odstraněn a problém musí být vyřešen před tím, než je zpráva vložena do zamýšlené fronty. Dále správce front změní pole *Format* deskriptoru zpráv (MQMD) tak, aby indikovalo, že zpráva obsahuje strukturu MQDLH.

Struktura MQDLH

Doporučuje se přidat strukturu MQDLH ke všem zprávám, které jste vložili do fronty nedoručených zpráv. Pokud však chcete použít obslužnou rutinu nedoručených zpráv poskytovanou některými produkty WebSphere MQ , ***musíte*** přidat strukturu MQDLH do vašich zpráv.

Přidání záhlaví do zprávy může pro frontu nedoručených zpráv příliš dlouhé, takže se vždy ujistěte, že jsou zprávy kratší než maximální velikost povolená pro frontu nedoručených zpráv, a to alespoň hodnotou konstanty MQ_MSG_HEADER_LENGTH. Maximální velikost zpráv povolených ve frontě je určena hodnotou atributu *MaxMsgLength* ve frontě. U fronty nedoručených zpráv se ujistěte, že tento atribut je

nastaven na maximum povolené správcem front. Pokud vaše aplikace nemůže doručit zprávu a zpráva je příliš dlouhá na to, aby byla vložena do fronty nedoručených zpráv, postupujte podle pokynů uvedených v popisu struktury MQDLH.

Ujistěte se, že je fronta nedoručených zpráv monitorována a že všechny zprávy přicházející do tohoto stavu se zpracují. Obslužná rutina fronty nedoručených zpráv je spouštěna jako dávkový obslužný program a lze ji použít k provádění různých akcí ve vybraných zprávách ve frontě nedoručených zpráv. Další podrobnosti viz “Zpracování fronty nedoručených zpráv” na stránce 533.

Je-li konverze dat nezbytná, převede správce front informace v záhlaví, když použijete volbu MQGMO_CONVERT na volání MQGET. Je-li proces, který zprávu vkládá, je MCA, je za záhlavím následován veškerý text původní zprávy.

Zprávy vkládané do fronty nedoručených zpráv mohou být zkráceny v případě, že jsou pro tuto frontu příliš dlouhé. Možným označením této situace jsou zprávy ve frontě nedoručených zpráv, které mají stejnou délku jako hodnota atributu *MaxMsgLength* ve frontě.

Zpracování fronty nedoručených zpráv

Tyto informace obsahují informace o programovacím rozhraní generální-použití při použití zpracování fronty nedoručených zpráv.

Zpracování fronty nedoručených zpráv závisí na požadavcích lokálního systému, ale při sestavování specifikace zvažte následující skutečnosti:

- Zpráva může být identifikována jako záhlaví fronty nedoručených zpráv, protože hodnota pole formátu v deskriptoru MQMD je MQFMT_DEAD_LETTER_HEADER.
- V produktu WebSphere MQ for z/OS pomocí CICS, pokud program MCA vloží tuto zprávu do fronty nedoručených zpráv, pole *PutApplType* je MQAT_CICS a pole *PutApplName* je *ApplId* systému CICS , za nímž následuje název transakce MCA.
- Příčina pro zprávu, která má být směrována do fronty nedoručených zpráv, je obsažena v poli *Reason* záhlaví fronty nedoručených zpráv.
- Hlavička fronty nedoručených zpráv obsahuje podrobnosti o názvu cílové fronty a názvu správce front.
- Hlavička fronty nedoručených zpráv obsahuje pole, která musí být obnovena v deskriptoru zpráv před tím, než je zpráva vložena do cílové fronty. Patří mezi ně:
  1. *Encoding*
  2. *CodedCharSetId*
  3. *Format*
- Deskriptor zprávy je stejný jako PUT původní aplikací, s výjimkou tří zobrazených polí (Encoding, CodedCharSetIda Format).

Vaše aplikace fronty nedoručených zpráv musí provést jednu nebo více z následujících možností:

- Prověřte pole *Reason* . Je možné, že zpráva byla vložena MCA z následujících důvodů:
  - Zpráva byla delší než maximální velikost zprávy pro kanál.

    Příčina: MQRC_MSG_TOO_BIG_FOR_CHANNEL
  - Zprávu nelze zařadit do cílové fronty.

    Důvodem je libovolný kód příčiny MQRC_ *, který může být vrácen operací MQPUT
  - Uživatelská procedura požadovala tuto akci

    Kód příčiny je dodán uživatelskou procedurou nebo výchozí hodnota MQRC_SUPPRESSED_BY_EXIT.
- Pokuste se předat zprávu do zamýšleného místa určení, pokud je to možné.
- Zachovejte zprávu po určitou dobu před vyřazením, je-li určena příčina zneužití, ale ne okamžitě opravitelná tabulka.
- Pokyny pro administrátory opravujte o problémech tam, kde je to určeno.
- Vyřaďte zprávy, které jsou poškozené nebo které nejsou zpracovatelné.

Existují dva způsoby, jak se vypořádat se zprávami, které jste se zotavili z fronty zablokovaných dopisů:

1. Je-li zpráva pro lokální frontu:
   - Provést všechny překlady kódu nezbytné pro extrakci dat aplikace
   - Provést převody kódu na těchto datech, pokud se jedná o lokální funkci
   - Vložení výsledné zprávy do lokální fronty se všemi podrobnostmi obnoveného deskriptoru zpráv
2. Je-li zpráva určena pro vzdálenou frontu, vložte ji do fronty.

Informace o tom, jak jsou obsluhovány nedoručené zprávy v distribuovaném prostředí s frontou, najdete v tématu Co se děje, když nelze zprávu doručit?.

Programování multicast

Tyto informace použijte k seznámení se s úlohami programování výběrového vysílání produktu WebSphere MQ , jako je připojení ke správci front a hlášení výjimek.

Výběrové vysílání produktu WebSphere MQ bylo navrženo tak, aby bylo co nejvíce transparentní pro uživatele a přesto je kompatibilní s existujícími aplikacemi. Definování objektu COMMINFO a nastavení parametrů **MCAST** a **COMMINFO** objektu TOPIC znamená, že existující aplikace produktu WebSphere MQ nevyžadují výrazné přepsání pro použití výběrového vysílání. Mohou však existovat určitá omezení (více informací viz “Výběrové vysílání a rozhraní fronty zpráv” na stránce 534) a některé bezpečnostní otázky (viz Zabezpečení výběrového vysílání , kde získáte další informace).

Výběrové vysílání a rozhraní fronty zpráv

Tyto informace vám pomohou pochopit hlavní koncepce MQI a informace o tom, jak souvisejí s výběrovým vysíláním produktu WebSphere MQ .

Odběry výběrového vysílání jsou netrvanlivé; protože nejsou k dispozici žádné fyzické fronty, neexistuje místo pro uložení zpráv offline, které byly vytvořeny trvalnými odběry.

Poté, co aplikace bude přihlášena k odběru tématu výběrového vysílání, je mu přidělen popisovač objektu, který může spotřebovat, nebo MQGET, jako by se jednalo o popisovač fronty. To znamená, že jsou podporovány pouze spravované odběry výběrového vysílání (odběry vytvořené pomocí MQSO_MANAGED), což znamená, že není možné vytvořit odběr a "bod" zprávy ve frontě. To znamená, že zprávy musí být spotřebovávány z manipulátoru objektu vráceného při volání odběru. Na straně klienta jsou zprávy uloženy do vyrovnávací paměti zpráv, dokud je klient nespotřebuje. Další informace naleznete v sekci MessageBuffer konfiguračního souboru klienta . Pokud klient nepodrží krok s rychlostí publikování, budou zprávy vyřazeny podle potřeby, přičemž nejstarší zprávy byly vyřazeny jako první.

Obvykle se jedná o rozhodnutí administrace, zda aplikace používá výběrové vysílání nebo ne, určené nastavením atributu MCAST objektu TOPIC. Pokud musí publikační aplikace zajistit, aby výběrové vysílání nebylo použito, může použít volbu `MQOO_NO_MULTICAST` . Podobně i odběratelská aplikace může zajistit, aby výběrové vysílání nebylo používáno při přihlašování k odběru s použitím volby `MQSO_NO_MULTICAST` .

Výběrové vysílání produktu WebSphere MQ podporuje použití selektorů zpráv. Selektor je používán aplikací k registraci svého zájmu pouze v těch zprávách s vlastnostmi, které splňují dotaz SQL92 , který představuje řetězec výběru. Další informace o selektorech zpráv viz “Selektory.” na stránce 21.

V následující tabulce jsou uvedeny všechny hlavní koncepce MQI a informace o tom, jak souvisí s výběrovým vysíláním:

Tabulka 71. Koncepce MQI a jejich souvislost s výběrovým vysíláním

Koncepce MQI	**Akce při pokusu pomocí výběrového vysílání**	**Kód příčiny**
Vložení zprávy o nulové délce	Odmítnuto	2005 (07D5) (RC2005): CHYBA MQRC_BUFFER_LENGTH_ERROR

Tabulka 71. Koncepce MQI a jejich souvislost s výběrovým vysíláním (pokračování)

Koncepce MQI	Akce při pokusu pomocí výběrového vysílání	Kód příčiny
Seskupení	Odmítnuto	2046 (07FE) (RC2046): CHYBA MQRC_OPTIONS_ERROR
Segmentace	Odmítnuto	2443 (098B) (RC2443): MQRC_SEGMENTATION_NOT_ALLOWED
Distribuční seznamy	Odmítnuto	2154 (086A) (RC2154): MQRC_REC_PRESENT_ERROR
MQINQ	Odmítnuto pro zpracování témat: MQINQ a MQSET témat nejsou podporovány.	2038 (07F6) (RC2038): MQRC_NOT_OPEN_FOR_INQUIRE
	Přijato pro spravovaný popisovač. Pouze `Aktuální hloubka` může být dotazovaná.	• Je-li hodnota Aktuální hloubka, pak neexistuje žádný vhodný kód příčiny. • Je-li hodnota jiná než Aktuální hloubka, kód příčiny je 2067 (0813) (RC2067): MQRC_SELECTOR_ERROR.
MQSET	Zamítnuto pro všechny popisovače.	2040 (07F8) (RC2040): MQRC_NOT_OPEN_FOR_SET
Transakce (XA nebo ne)	Odmítnuto	2072 (0818) (RC2072): MQRC_SYNCPOINT_NOT_AVAILABLE
Procházení zpráv	Odmítnuto	2036 (07F4) (RC2036): MQRC_NOT_OPEN_FOR_BROWSE
Uzamknout zprávy	Odmítnuto	2046 (07FE) (RC2046): CHYBA MQRC_OPTIONS_ERROR
Procházet značkou	Odmítnuto	2036 (07F4) (RC2036): MQRC_NOT_OPEN_FOR_BROWSE
Předat kontext	Odmítnuto	2046 (07FE) (RC2046): CHYBA MQRC_OPTIONS_ERROR
MQPUT1	Zamítnuto. Je neplatné pokusit se MQPUT1 použít pouze pro téma Výběrové vysílání.	2560 (0A00) (RC2560): MQRC_MULTICAST_ONLY
trvalý odběr	Zamítnuto, je-li téma označeno jako "Pouze výběrové vysílání", jinak se provede odběr jiného typu než Výběrové vysílání.	2436 (0984) (RC2436): MQRC_DURABILITY_NOT_ALLOWED

Tabulka 71. Koncepce MQI a jejich souvislost s výběrovým vysíláním (pokračování)		
Koncepce MQI	**Akce při pokusu pomocí výběrového vysílání**	**Kód příčiny**
TopicString > 255	Zamítnuto. Je-li řetězec tématu delší než 255 znaků, je odmítnut v klientovi.	2425 (0979) (RC2425): MQRC_TOPIC_STRING_ERROR
Provedené nespravované odběry	Zamítnuto, je-li téma označeno jako "Pouze výběrové vysílání", jinak se provede odběr jiného typu než Výběrové vysílání.	2046 (07FE) (RC2046): CHYBA MQRC_OPTIONS_ERROR
MQPMO_NOT_OWN_SUBS	Odmítnuto	2046 (07FE) (RC2046): CHYBA MQRC_OPTIONS_ERROR

Následující položky se rozbalí v některých koncepcích MQI z předchozí tabulky a poskytují informace o některých konceptech MQI, které nejsou v tabulce:

Trvalost zpráv
V případě odběratelů výběrového vysílání jsou trvalé zprávy od vydavatele doručovány neobnovitelným způsobem.

Zkrácení zprávy
Oříznutí zprávy je podporováno, což znamená, že je možné, aby aplikace mohla:

1. Zadejte příkaz MQGET.
2. Získání objektu MQRC_TRUNCATED_MSG_FAILED.
3. Přidělte větší vyrovnávací paměť.
4. Znovu zadejte příkaz MQGET a načtěte zprávu.

Vypršení platnosti odběru
Vypršení platnosti odběru není podporováno. Jakýkoli pokus o nastavení vypršení platnosti je ignorován.

Vysoká dostupnost pro výběrové vysílání

Tyto informace použijte k pochopení nepřetržitého provozu výběrového vysílání produktu WebSphere MQ . Přestože se produkt WebSphere MQ připojuje ke správci front produktu WebSphere MQ , zprávy neprocházejí tímto správcem front.

Přestože musí být vytvořeno připojení ke správci front, aby bylo možné objekt tématu výběrového vysílání MQOPEN nebo MQSUB MQUB, neprotékat prostřednictvím správce front zprávy samotné. Proto je po dokončení operace MQOPEN nebo MQSUB na objektu tématu výběrového vysílání možné pokračovat ve vysílání zpráv výběrového vysílání, a to i v případě, že připojení ke správci front bylo ztraceno. Existují dva režimy provozu:

Je vytvořeno běžné připojení ke správci front
Komunikaci multicast je možné, když existuje připojení ke správci front. Pokud připojení selže, jsou použita běžná pravidla MQI, například; příkaz MQPUT pro obsluhu objektů výběrového vysílání vrátí hodnotu 2009 (07D9) (RC2009): MQRC_CONNECTION_BROKEN.

Připojuje se znovu připojení klienta ke správci front.
Komunikace multicast je možná i během reconnection cyklu. To znamená, že i když došlo k přerušení připojení ke správci front, tím není ovlivněno vložení zpráv výběrového vysílání a příjem zpráv výběrového vysílání. Klient se pokusí znovu připojit ke správci front, a pokud toto opětovné připojení selže, stane se popisovač připojení přerušený a všechna volání MQI, včetně těch s výběrovým vysíláním, selžou. Další informace viz: Automatické opětovné připojení klienta

Pokud některá aplikace explicitně vydá příkaz MQDISC, budou všechny odběry výběrového vysílání a manipulátory objektů zavřeny.

Kontinuální nepřetržitá operace typu

Jednou z výhod komunikace typu P2P mezi klienty je to, že zprávy nemusí procházet správcem front. Pokud se tedy připojení ke správci front přeruší, bude přenos zpráv pokračovat. Pro požadavky na souvislé zprávy v tomto režimu se vztahují následující omezení:

- Připojení musí být vytvořeno pomocí jedné z voleb MQCNO_RECONICONNECT_ * pro průběžnou operaci. Tento proces znamená, že ačkoli komunikační relace může být porušena, není aktuální manipulátor připojení přerušený a nachází se v novém připojeném stavu. Dojde-li k selhání nového připojení, je nyní popisovač připojení přerušen, což brání všem dalším voláním MQI.
- V tomto režimu jsou podporovány pouze příkazy MQPUT, MQGET, MQINQ a Async Consume. Příkazové příkazy MQOPEN, MQCLOSE nebo MQDISC vyžadují opětovné připojení ke správci front, aby bylo dokončeno.
- Stav toků ke správci front se zastaví; jakýkoli stav ve správci front proto může být zastaralý nebo chybějící. To znamená, že klienti mohou odesílat a přijímat zprávy a že ve správci front není znám žádný stav. Další informace naleznete v tématu: Monitorování aplikace výběrového vysílání

Převod dat v rozhraní MQI pro systém zpráv výběrového vysílání

Pomocí těchto informací můžete pochopit, jak převod dat pracuje pro systém zpráv výběrového vysílání WebSphere MQ Multicast.

WebSphere MQ Multicast je sdílený, bezspojový protokol, a proto není možné, aby každý klient mohl provádět specifické požadavky na konverzi dat. Každý klient přihlášený k odběru stejného toku výběrového vysílání přijímá stejná binární data; proto, je-li zapotřebí konverze dat produktu WebSphere MQ , provede se konverze lokálně na každém klientovi.

Data jsou převedena na klienta pro provoz výběrového vysílání produktu WebSphere MQ . Je-li zadána volba **MQGMO_CONVERT** , převod dat se provádí podle požadavku. Uživatelsky definované formáty potřebují uživatelskou proceduru pro převod dat nainstalovanou na klientovi; informace o tom, které knihovny jsou nyní v balících klienta a serveru, najdete v příručce “Zápis uživatelských procedur pro převod dat” na stránce 399 .

Informace o administraci převodu dat naleznete v tématu Povolení převodu dat pro systém zpráv výběrového vysílání.

Další informace o konverzi dat najdete v tématu Převod dat.

Další informace o uživatelských procedurách pro převod dat a produktu `ClientExitPath`naleznete v části ClientExitPath stanza konfiguračního souboru klienta.

Vykazování výjimek výběrového vysílání

Tyto informace použijte ke zjištění informací o obslužných rutinách událostí výběrového vysílání produktu WebSphere MQ a hlášení výjimek výběrového vysílání WebSphere MQ .

Výběrové vysílání produktu WebSphere MQ pomáhá s určováním problémů voláním obslužné rutiny událostí za účelem hlášení událostí výběrového vysílání, které jsou ohlášeny za použití standardního mechanizmu obslužné rutiny událostí produktu WebSphere MQ .

Jednotlivé události výběrového vysílání mohou mít za následek vyvolání více než jedné události produktu WebSphere MQ , protože může existovat více obslužných rutin připojení produktu `MQHCONN` používajících

stejný vysílač výběrového vysílání nebo příjemce. Každá výjimka výběrového vysílání však způsobí, že bude volána pouze jedna obslužná rutina událostí pro připojení WebSphere MQ .

Konstanta produktu WebSphere MQ MQCBDO_EVENT_CALL umožňuje aplikacím registrovat zpětné volání pouze pro příjem událostí produktu WebSphere MQ a aplikace MQCBDO_MC_EVENT_CALL umožňuje registraci zpětného volání pro příjem pouze událostí výběrového vysílání. Jsou-li použity obě konstanty, jsou přijaty oba typy událostí.

Vyžádání událostí výběrového

Události výběrového vysílání produktu WebSphere MQ používají konstantu MQCBDO_MC_EVENT_CALL v poli cbd.Options . Následující příklad ukazuje, jak požadovat události výběrového vysílání:

```
cbd.CallbackType     = MQCBT_EVENT_HANDLER;
cbd.Options          = MQCBDO_MC_EVENT_CALL;
cbd.CallbackFunction = EventHandler;
MQCB(Hcon,MQOP_REGISTER,&cbd,MQHO_UNUSABLE_HOBJ,NULL,NULL,&CompCode,&Reason);
```

Je-li zadána volba MQCBDO_MC_EVENT_CALL pro pole cbd.Options , obslužná rutina událostí odešle místo událostí na úrovni připojení pouze události výběrového vysílání WebSphere MQ . Pro požadavek, aby byly do obslužné rutiny událostí odeslány oba typy událostí, musí aplikace zadat konstantu MQCBDO_EVENT_CALL do pole cbd.Options stejně jako konstantu MQCBDO_MC_EVENT_CALL , jak ukazuje následující příklad:

```
cbd.CallbackType     = MQCBT_EVENT_HANDLER;
cbd.Options          = MQCBDO_EVENT_CALL | MQCBDO_MC_EVENT_CALL
cbd.CallbackFunction = EventHandler;
MQCB(Hcon,MQOP_REGISTER,&cbd,MQHO_UNUSABLE_HOBJ,NULL,NULL,&CompCode,&Reason);
```

Není-li použit ani jeden z těchto konstant, budou do obslužné rutiny událostí odeslány pouze události úrovně připojení.

Další informace o hodnotách pro pole Options naleznete v části Volby (MQLONG).

Formát událostí výběrového vysílání

Výjimky výběrového vysílání produktu WebSphere MQ obsahují některé podpůrné informace, které jsou vráceny v argumentu **Buffer** funkce zpětného volání. Ukazatel **Buffer** ukazuje na pole ukazatelů a v poli MQCBC.DataLength je uvedena velikost pole v bajtech. První prvek pole vždy ukazuje na krátký textový popis události. Další parametry mohou být dodány v závislosti na typu události. V následující tabulce jsou uvedeny výjimky:

Tabulka 72. Popisy kódů událostí výběrového vysílání

Kód události	Popis	Další data
SRÁŽKA MQMTEV_PACKET_LOSS	Neopravitelná ztráta paketů	Počet ztracených paketů
ČASOVÝ LIMIT MQMCAV_HEARTBEAT_TIMEOUT	Dlouhá absence řídicího paketu prezenčního signálu	Není k dispozici
KONFLIKT MQMTEV_VERSION_CONFLICT	Přijímání novějších paketů verze protokolu	Není k dispozici
MQMCEVA_RELIABILITY	Různé režimy spolehlivosti vysílače a přijímače	Není k dispozici
UZAVŘÍT MQMCEV_CLOSED_TRANS	Přenos tématu je uzavřen jedním zdrojem	Není k dispozici
CHYBA MQMTEV_STREAM_ERROR	Zjištěna chyba v proudu	Není k dispozici
ZDROJ MQMCEV_NEW_SOURCE	Nový zdroj začne vysílat na téma	Zdrojová struktura

Tabulka 72. Popisy kódů událostí výběrového vysílání (pokračování)

Kód události	Popis	Další data
MAČ_PŘÍJEMCE_PŘIJMU_MQUT_DATA_OŘÍZNUTO	Odebrané pakety z hodnoty PacketQ kvůli vypršení času nebo vypršení platnosti prostoru.	Počet ořízených paketů
SKONČIT PLATNOST: MQMCEV_PACKET_LOSS_NICK_EXPIRE	Neopravitelná ztráta paketu kvůli vypršení platnosti NACK	Počet ztracených paketů
BYL PŘEKROČEN PARAMETR MQMVAR_ACK_RETRIES_	Pakety odebrané z historie po překročení **max_ack_retries**	Počet odebraných paketů
MQMCEV_STREAM_SUSPEND_NACK	NACK byly pozastaveny na proud akceptovaný tímto tématem	Pozastavit ID proudu
		Doba, po kterou je proud pozastaven
MQMCEV_STREAM_RESUME_NACK	NACKs byly obnoveny poté, co byly pozastaveny na proudu	ID proudu
VYLOUČENÝ PROUD MQMTEV_STREAM_EX	Proud přijatý tímto tématem byl zamítnut v důsledku požadavku na vyloučení	ID proudu
ZPRÁVA MQMTEV_FIRST_MESSAGE	První zpráva ze zdroje	Číslo zprávy.
SELHÁNÍ FUNKCE MQMCEV_LACE_JOIN_FAILURE	Spuštění relace zpožděného spojení se nezdařilo	Není k dispozici
SKRYTÍ MQMEV_MESSAGE_LOSS	Neopravitelná ztráta zpráv	Počet ztracených zpráv
SELHÁNÍ MQMTEV_SEND_PACKET_FAILURE	Vysílač výběrového vysílání selhal při odesílání paketu výběrového vysílání	Není k dispozici
ZPOŽDĚNÍ MQMEV_REPACE_	Přijímač výběrového vysílání neobdržel paket opravy pro neprovedený NAK	Není k dispozici
MQMCEV_MEMORY_ALERT_ON	Vyrovnávací paměti příjmu zásobníku se zaplňují	Procento využití fondu vyrovnávacích pamětí
MQMCEV_MEMORY_ALERT_OFF	Vyrovnávací paměti příjmu zásobníku jsou mimo provoz.	Procento využití fondu vyrovnávacích pamětí
MQMCE_V_NACK_ALERT_ON	Rychlost požadavku na opravu paketu byla dosažena horní mez	Aktuální četnost požadavků na opravu v paketech za sekundu
MQMCEV_NACK_ALERT_OFF	Rychlost požadavků paketů na opravu zásobníku je mimo provoz na normální úroveň	Aktuální četnost požadavků na opravu v paketech za sekundu
MQMETEV_REPACE_ALERT_ON	Rychlost odesílání paketů pro opravu paketu pro opravu vysílače dosáhla horní meze	Není k dispozici
MQMCEV_REPAIR_ALERT_OFF	Rychlost odeslání paketu pro opravu vysílače je mimo provoz.	Není k dispozici
MQMTEV_SHM_DEST_UNUSABLE	Oblast sdílené paměti použitá místem určení tématu vysílače byla zjištěna jako nepoužitelná.	Není k dispozici

Tabulka 72. Popisy kódů událostí výběrového vysílání (pokračování)		
Kód události	**Popis**	**Další data**
MQMTEV_SHM_PORT_UNUSABLE	Port sdílené paměti použitý instancí příjemce byl zjištěn jako nepoužitelný	Není k dispozici
SELHÁNÍ MQMCEV_CCT_GETTIME_FAILED	Čas získání z koordinovaného času klastru se nezdařil	Není k dispozici
SELHÁNÍ PŘI SELHÁNÍ MQMTEV_DEST_INTERFACE_FAILURE	Síťové rozhraní použité místem určení tématu vysílače se nezdařilo a záložní síťové rozhraní je nedostupné.	
MQMCEV_DEST_INTERFACE_FAILOVER	Síťové rozhraní použité místem určení tématu vysílače se nezdařilo a úspěšné překonání selhání na jiné rozhraní bylo dokončeno.	
MQMTEV_PORT_INTERFACE-SELHÁNÍ	Síťové rozhraní použité příjemcem rmmPort se nezdařilo a záložní síťové rozhraní je nedostupné (nebo se také nezdařilo)	Konfigurace RMM
MQMCEV_PORT_INTERFACE_FAILOVER	Síťové rozhraní použité příjemcem rmmPort se nezdařilo a úspěšné překonání selhání na jiné rozhraní bylo dokončeno	Konfigurace RMM

Použití .NET

Třídy WebSphere MQ pro prostředí .NET umožňují programu napsaným v programovacím rámci .NET pro připojení k produktu WebSphere MQ jako klienta WebSphere MQ MQI nebo k přímému připojení k serveru WebSphere MQ .

Máte-li aplikace, které používají Microsoft's .NET Framework a chcete využívat výhody produktu WebSphere MQ, musíte použít třídy WebSphere MQ pro prostředí .NET.

Objektově objektově orientované rozhraní WebSphere MQ .NET se liší od rozhraní MQI v tom, že používá spíše metody objektů než při použití příkazových slov MQI.

Procedurální programovací rozhraní produktu WebSphere MQ je založeno na příkazových slovech, jako jsou např. v následujícím seznamu:

```
MQCONN, MQDISC, MQOPEN, MQCLOSE,
MQINQ, MQSET, MQGET, MQPUT, MQSUB
```

Tato příkazová slova slouží jako parametr k obsluze objektu WebSphere MQ , na kterém mají být provozovány. Vzhledem k tomu, že prostředí .NET je objektově orientované, toto kolo mění programovací rozhraní .NET. Váš program se skládá ze sady objektů produktu WebSphere MQ , které se při volání metod na těchto objektech chovají. Programy můžete psát v libovolném jazyce, který podporuje prostředí .NET.

Když použijete procedurální rozhraní, odpojíte se od správce front pomocí volání MQDISC (*Hconn*, CompCode, Reason), kde *Hconn* je manipulátor pro správce front.

V rozhraní .NET je správce front reprezentován objektem třídy MQQueueManager. Odpojení od správce front voláte voláním metody Disconnect () na dané třídě.

```
// declare an object of type queue manager
```

```
MQQueueManager queueManager=new MQQueueManager();
...
// do something...
...
// disconnect from the queue manager
queueManager.Disconnect();
```

Třídy WebSphere MQ pro prostředí .NET jsou sadou tříd, které umožňují interakci aplikací .NET s produktem WebSphere MQ. Představují různé komponenty produktu WebSphere MQ , které vaše aplikace používá, jako jsou správce front, fronty, kanály a zprávy. Podrobné informace o těchto třídách naleznete v tématu Třídy a rozhraní produktu WebSphere MQ .NET.

Než budete moci zkompilovat všechny aplikace, které napíšete, musíte mít nainstalovanou platformu .NET Framework. Pokyny k instalaci tříd produktu WebSphere MQ pro prostředí .NET a .NET Framework naleznete v tématu “Instalace tříd produktu WebSphere MQ pro prostředí .NET” na stránce 542.

Související pojmy

Technický přehled

“Volby připojení” na stránce 541
K dispozici jsou tři režimy připojení tříd WebSphere MQ pro prostředí .NET ke správci front. Zvažte, který typ připojení nejlépe vyhovuje vašim požadavkům.

“Použití tříd produktu WebSphere MQ pro prostředí .NET” na stránce 552
Tato kolekce témat popisuje, jak nakonfigurovat systém ke spuštění ukázkových programů pro ověření instalace tříd WebSphere MQ pro instalaci .NET a o tom, jak spouštět vlastní programy.

“Řešení problémů s produktem WebSphere MQ .NET” na stránce 555
Pokud se program nedokončí úspěšně, spusťte jednu z ukázkových aplikací a postupujte podle doporučení uvedených v diagnostických zprávách.

“Psaní a implementace programů WebSphere MQ .NET” na stránce 555
Chcete-li používat třídy WebSphere MQ pro .NET pro přístup k frontám produktu WebSphere MQ , můžete psát programy v libovolném jazyce, který je podporován rozhraním .NET, který obsahuje zprávy pro vkládání zpráv do front zpráv WebSphere MQ a jejich získání od nich.

“Vlastní kanál produktu IBM WebSphere MQ pro produkt Microsoft Windows Communication Foundation (WCF)” na stránce 575
Vlastní kanál produktu Microsoft Windows Communication Foundation (WCF) pro produkt IBM WebSphere MQ odesílá a přijímá zprávy mezi klienty WCF a službami.

“Rozhodování o tom, jaký programovací jazyk použít” na stránce 75
Tyto informace použijte k vyhledání informací o programovacích jazycích a rámcích podporovaných produktem IBM WebSphere MQa o některých aspektech jejich použití.

“Vývoj aplikací” na stránce 7
Produkt IBM WebSphere MQ nabízí několik způsobů, jak vyvíjet aplikace k odesílání a přijímání zpráv, které potřebujete k podpoře vašich obchodních procesů. Také můžete vyvíjet aplikace pro správu správců front a souvisejících prostředků.

Začínáme s třídami produktu WebSphere MQ pro prostředí .NET

Třídy WebSphere MQ pro prostředí .NET umožňují programu napsaným v programovacím rámci .NET pro připojení k produktu WebSphere MQ jako klienta WebSphere MQ MQI nebo k přímému připojení k serveru WebSphere MQ .

Volby připojení

K dispozici jsou tři režimy připojení tříd WebSphere MQ pro prostředí .NET ke správci front. Zvažte, který typ připojení nejlépe vyhovuje vašim požadavkům.

Připojení vazeb klienta

Chcete-li používat třídy WebSphere MQ pro prostředí .NET jako klient WebSphere MQ MQI, můžete jej nainstalovat pomocí klienta WebSphere MQ MQI, a to buď na počítači serveru WebSphere MQ , nebo na samostatném počítači. Připojení vazeb klienta může využívat transakce XA nebo non-XA.

Připojení vazeb serveru

Při použití v režimu vazeb serveru třídy produktu WebSphere MQ používají rozhraní API správce front místo komunikace prostřednictvím sítě rozhraní API správce front. To poskytuje lepší výkon pro aplikace WebSphere MQ než použití síťových připojení.

Chcete-li používat připojení vazeb, je třeba instalovat třídy produktu WebSphere MQ pro prostředí .NET na serveru WebSphere MQ .

Připojení spravovaného klienta

Připojení vytvořené v tomto režimu se připojuje jako klient WebSphere MQ k serveru WebSphere MQ spuštěnému buď na lokálním nebo vzdáleném počítači.

Třídy produktu WebSphere MQ pro připojení .NET v tomto režimu zůstávají ve spravovaném kódu .NET a nevolají nativní služby. Další informace o spravovaném kódu najdete v dokumentaci společnosti Microsoft .

Pro použití spravovaného klienta existuje několik omezení. Další informace o těchto viz “Připojení spravovaného klienta” na stránce 556.

Instalace tříd produktu WebSphere MQ pro prostředí .NET

Třídy WebSphere MQ pro prostředí .NET, včetně ukázek, jsou instalovány spolu s produktem WebSphere MQ. Je zde předpoklad prostředí Microsoft .NET Framework.

Nejnovější verze produktu WebSphere MQ tříd pro prostředí .NET je standardně nainstalována jako součást standardní instalace produktu WebSphere MQ v rámci funkce *Java and .NET Messaging and Web Services* . Pokyny k instalaci naleznete v tématu Instalace serveru IBM WebSphere MQ v systému Windows nebo Instalace klienta IBM WebSphere MQ v systémech Windows .

Pokud jste v prostředí s více instalačními produkty nainstalovali třídy WebSphere MQ pro prostředí .NET jako balík podpory, nemůžete instalovat produkt WebSphere MQ , pokud nejprve neodinstalujete balík podpory. Třídy WebSphere MQ pro funkci .NET, které jsou instalovány spolu s produktem WebSphere MQ , obsahují stejné funkce jako balík podpory.

K dispozici jsou také ukázkové aplikace, včetně zdrojových souborů, viz “Ukázkové aplikace” na stránce 553.

Chcete-li spustit třídy WebSphere MQ pro prostředí .NET na 32bitových nebo 64bitových platformách, musíte mít nainstalován produkt Microsoft .NET Framework V2.0 nebo vyšší.

Poznámka: Není-li před instalací produktu WebSphere MQ V7.0.1nainstalován produkt Microsoft .NET Framework v2.0 nebo vyšší, bude instalace produktu WebSphere MQ bez chyby pokračovat, ale třídy WebSphere MQ pro prostředí .NET nebudou k dispozici. Je-li prostředí .NET Framework nainstalováno po instalaci produktu WebSphere MQ 7.0.1, musí být sestavení produktu WebSphere .NET registrována spuštěním skriptu *WMQInstallDir*`\bin\amqiRegisterdotNet.cmd` , kde *WMQInstallDir* je adresář, do kterého je nainstalován produkt WebSphere MQ 7.0.1 . Tento skript nainstaluje vyžadované sestavení v mezipaměti GAC (Global Assembly Cache). Sada souborů `amqi*.log` zaznamenávající provedené akce se vytvoří v adresáři %TEMP% .

Informace o použití vlastního kanálu produktu WebSphere MQ pro produkt Microsoft WCF s prostředím .NET 3 naleznete v tématu: “Vlastní kanál produktu IBM WebSphere MQ pro produkt Microsoft Windows Communication Foundation (WCF)” na stránce 575 .

Distribuované transakce v .NET

Distribuované transakce nebo globální transakce umožňují klientským aplikacím zahrnout do jedné transakce několik různých zdrojů dat na dvou nebo více síťových systémech.

V distribuovaných transakcích správce transakcí koordinuje a spravuje transakci mezi dvěma či více správci prostředků.

Transakce mohou být buď jednofázové, nebo dvoufázové procesy vázaného zpracování. Jednofázové potvrzení je proces, jehož jediným správcem prostředků je zapojen do procesu transakce a dvoufázového potvrzování, je-li v transakci více než jeden správce prostředků, který se podílí na transakci. Ve dvoufázovém procesu potvrzování odešle správce transakcí připravenou výzvu ke kontrole, zda jsou všichni správci prostředků připraveni k potvrzení. Když přijme potvrzení od všech správců prostředků, vydá se výzva k potvrzení. V opačném případě dojde k odvolání transakce při celé transakci. Další podrobnosti viz Správa transakcí a podpora . Správci prostředků by měli informovat správce transakcí o jejich účasti v transakci. Když správce prostředků informuje správce transakcí o své účasti, získá správce prostředků zpětné volání od správce transakcí, když transakce potvrdí nebo vrátí zpět.

Třídy WebSphere MQ .NET již podporují distribuované transakce v nespravovaných připojeních a v režimu vazeb serveru. V těchto režimech produkt WebSphere MQ .NET Classes deleguje všechna svá volání na klienta rozšířených transakcí jazyka C, který spravuje zpracování transakcí v zastoupení .NET.

WebSphere MQ.NET nyní podporují distribuované transakce ve spravovaném režimu, kde produkt WebSphere MQ .NET Classes používá obor názvů System.Transactions pro podporu distribuovaných transakcí. Infrastruktura System.Transactions usnadňuje a zefektivňuje transakční programování tím, že podporuje transakce zahájené ve všech správcích prostředků včetně produktu WebSphere MQ. Aplikace .NET produktu WebSphere MQ může vkládat a získávat zprávy pomocí programování implicitních transakcí prostředí .NET nebo explicitního modelu programování transakcí. V implicitních transakcích jsou hranice transakce vytvářeny aplikačním programem, který rozhoduje o tom, kdy se má potvrdit, odvolat (pro explicitní transakce) nebo dokončit transakci. V explicitních transakcích musíte výslovně uvést, zda chcete potvrdit, odvolat a dokončit transakci.

WebSphere MQ.NET používá jako správce transakcí Microsoft distributed transaction coordinator (MS DTC), který koordinuje a spravuje transakci mezi více správci prostředků. Produkt WebSphere MQ se používá jako správce prostředků.

WebSphere MQ.NET postupuje podle modelu X/Open Distributed Transaction Processing (DTP). Model distribuovaného zpracování transakcí X/Open je distribuovaným modelem zpracování transakcí navrženým skupinou Open Group, konsorciem dodavatele. Tento model je standardem většiny komerčních dodavatelů v transakčním zpracování transakcí a v doménách databáze. Většina komerčních produktů pro správu transakcí podporuje model X/DTP.

Způsoby transakce

- “Distribuované transakce ve spravovaném režimu” na stránce 544
- Distribuované transakce pro nespravovaný režim

Koordinace transakcí v různých scénářích

- Připojení se může podílet na několika transakcích, ale v libovolném časovém okamžiku je aktivní pouze jedna transakce.
- Během transakce je to MQQueueManager.Volání odpojení bylo uznáno. V tomto případě je transakce požádána o odvolání transakce.
- V průběhu transakce je volání MQQueue.Close nebo MQTopic.Close uznáno. V tomto případě je transakce požádána o odvolání transakce.
- Hranice transakce jsou vytvářeny aplikačním programem, který rozhoduje o tom, kdy se má potvrdit, odvolat (pro explicitní transakce) nebo dokončit transakci (pro implicitní transakce).

- Pokud se aplikace klienta během transakce přeruší s neočekávanou chybou před zadáním volání příkazu Put nebo Get ve volání fronty nebo tématu, transakce je odvolána a dojde k vyvolání výjimky MQException.
- Je-li kód příčiny MQCC_FAILED vrácen během volání funkce Put nebo Get v rámci volání fronty nebo tématu, dojde k výjimce MQException s kódem příčiny a transakce se bude přehrávány. Pokud správce transakcí již vydal připravovací volání, produkt WebSphere MQ .NET vrátí požadavek na přípravu nuceně odvolání transakce. Poté správce transakcí DTC vyvolá odvolání změn v aktuální práci se všemi správci prostředků v aktuálních okolních transakcích.
- Pokud během transakce zahrnující více správců prostředků způsobí, že některý z důvodů ochrany životního prostředí způsobí, že se operace Put nebo Get budou pozastaveny na dobu neurčitou, bude správce transakcí čekat až do stanoveného času. Po vypršení časového limitu dojde k odvolání veškeré aktuální práce se všemi správci prostředků v aktuálních okolních transakcích. Pokud toto čekání na dobu neurčitou nastane během fáze přípravy, správce transakcí může vypršet nebo může vyvolat nejisté volání na prostředku v takovém případě, že transakce je odvolána.
- Aplikace používající transakce musí vložit nebo získat zprávy pod SYNC_POINT. Je-li v rámci transakčního kontextu, který není pod názvem SYNC_POINT, vydána zpráva s voláním funkce Put nebo Get, volání selže s kódem příčiny MQRC_UNIT_OF_WORK_NOT_STARTED.

Rozdíly v chování mezi podporou spravovaných a nespravovaných klientských transakcí s použitím oboru názvů Microsoft .NET System.Transactions

Vnořené transakce mají objekt TransactionScope uvnitř jiného TransactionScope

- Plně spravovaný klient WebSphere .NET MQ .NET podporuje vnořené TransactionScope
- Nespravovaný klient WebSphere MQ .NET nepodporuje vnořené TransactionScope

Závislé transakce z System.Transactions

- Plně spravovaný klient WebSphere MQ .NET podporuje prostředek závislých transakcí poskytovaný produktem System.Transactions.
- WebSphere MQ .NET nespravovaný klient nepodporuje poskytovanou službu transakcí poskytovanou System.Transactions.

Ukázky produktu

Nové ukázky produktů `SimpleXAPuta` `SimpleXAGet` jsou dostupné v části `WebSphere MQ\tools\dotnet\samples\cs\base` . Ukázky jsou C# aplikace, které demonstrují pomocí MQPUT a MQGET pod Distribuovanými transakcemi pomocí oboru názvů SystemTransactions . Další informace o těchto ukázkách naleznete v tématu “Vytváření jednoduchých vložení a získání zpráv v rámci TransactionScope” na stránce 547 .

Distribuované transakce ve spravovaném režimu

Třídy WebSphere MQ .NET používají obor názvů System.Transactions pro podporu distribuovaných transakcí ve spravovaném režimu. Ve spravovaném režimu koordinuje MS DTC souřadnice a spravuje distribuované transakce na všech serverech uvedených v transakci.

Třídy WebSphere MQ .NET poskytují explicitní programovací model založený na třídě System.Transactions.Transaction a implicitním programovacím modelu za použití System.Transactions.TransactionScope, třída, kde jsou transakce automaticky spravovány infrastrukturou.

Implicitní transakce

Následující část kódu popisuje způsob, jakým aplikace WebSphere MQ .NET vloží zprávu pomocí programování implicitních transakcí prostředí .NET.

```
Using (TransactionScope scope = new TransactionScope ())
{
    Q.Put (putMsg,pmo);
    scope.Complete ();
}
```

```
Q.close();
qMgr.Disconect();}
```

Vysvětlení toku kódu implicitní transakce

Kód vytvoří objekt *TransactionScope* a vloží zprávu do rozsahu. Pak zavolá *Dokončit* , aby informoval koordinátor transakcí o dokončení transakce. Koordinátor transakcí nyní vydá *prepare* a *commit* (potvrdit) k dokončení transakce. Pokud je zjištěn problém, volá se *odvolání* .

Explicitní transakce

Následující kód popisuje, jak aplikace .NET produktu WebSphere MQ vkládá zprávy pomocí explicitního modelu programování transakcí .NET.

```
MQQueueManager qMgr = new MQQueuemanager ("MQQM);
MQQueue Q = QMGR.AccessQueue("Q", MQC.MQOO_OUTPUT+MQC.MQOO_INPUT_SHARED);
MQPutMessageOptions pmo = new MQPutMessageOptions();
pmo.Options = MQC.MQPMO_SYNCPOINT;
MQMessage putMsg1 = new MQMessage();
Using(CommittableTransaction tx = new CommittableTransaction()){
Transaction.Current = tx;
    try
    {
    Q.Put(MSG,pmo);
    tx.commit();
    }
    catch(Exception)
    {tx.rollback();}
    }

Q.close();
qMgr.Disconnect();
}
```

Vysvětlení toku kódu explicitní transakce

Část kódu vytvoří transakci pomocí třídy *CommitableTransaction* . Do této oblasti se vloží zpráva a pak při dokončení transakce explicitně volá příkaz *commit* (potvrdit). Jsou-li volány nějaké problémy *rollback* .

Distribuované transakce v nespravovaném režimu

WebSphere MQ.NET podporují nespravovaná připojení (klient) s použitím rozšířeného klienta transakcí a COM + /MTS jako koordinátora transakcí s použitím implicitního nebo explicitního modelu programování transakcí. V nespravovaném režimu delegují třídy produktu WebSphere MQ .NET všechny své volání na klienta rozšířených transakcí jazyka C, který spravuje zpracování transakcí pro prostředí .NET.

Zpracování transakce je řízeno externím správcem transakcí a koordinuje globální jednotku práce pod kontrolou rozhraní API správce transakcí. Přísloví MQBEGIN, MQCMIT a MQBACK jsou nedostupné. Třídy WebSphere MQ .NET vystavují tuto podporu prostřednictvím svého nespravovaného režimu přenosu (klient jazyka C). Viz téma Konfigurace správců transakcí standardu XA

MTS se vyvinul jako systém zpracování transakcí (TP), který poskytuje stejné funkce v systému Windows NT , jaký je k dispozici v CICS, Tuxedo, a na jiných platformách. Při instalaci serveru MTS se do systému Windows NT přidá samostatná služba, která se nazývá Microsoft Distributed Transaction Coordinator (MSDTC). MSDTC koordinuje transakce, které zahrnují samostatná datová úložiště nebo prostředky. Pro práci vyžaduje každé datové úložiště pro implementaci svého vlastního proprietárního správce prostředků.

WebSphere MQ je kompatibilní s MSDTC implementací rozhraní (proprietární rozhraní správce prostředků), které umožňuje mapování volání DTC XA na volání produktu WebSphere MQ(X/Open). Produkt WebSphere MQ hraje roli správce prostředků.

Když komponenta jako COM + požaduje přístup k produktu WebSphere MQ, COM obvykle kontroluje odpovídající objekt kontextu MTS, pokud je požadována transakce. Je-li požadována transakce, informuje COM diagnostický chybový kód DTC a automaticky spustí pro tuto operaci integrální transakci WebSphere MQ . Poté bude COM pracovat s daty pomocí softwaru MQMTS, vkládat a dostávat zprávy podle potřeby. Instance objektu získaná z COM volá metodu SetComplete nebo SetAbort poté, co všechny akce na datech jsou u konce. Když aplikace vydá příkaz SetComplete, volání signalizuje kód DTC, že aplikace

dokončila transakci a kód DTC může pokračovat v procesu dvoufázového potvrzování. DTC pak vyvolá volání MQMTS, která v řadě vyvolá volání WebSphere MQ , aby transakci potvrzovali nebo odvrátili.

Psaní aplikace WebSphere MQ .NET pomocí nespravovaného klienta

Chcete-li spustit v kontextu COM +, musí třída .NET zdědit ze systému.EnterpriseServices.ServicedComponent. Pravidla a doporučení pro vytváření sestav, které používají obsluhované komponenty, jsou následující:

Poznámka: Následující kroky jsou relevantní pouze v případě, že používáte režim System.EnterpriseServices .

- Třída a metoda spouštěná v COM + musí být veřejné (žádné vnitřní třídy a žádné chráněné nebo statické metody).
- Atributy třídy a metody: Atribut TransactionOption určuje úroveň transakce třídy, tj. zda jsou transakce zakázány, podporovány nebo vyžadovány. Atribut AutoComplete na metodě ExecuteUOW() instruuje COM +, aby potvrdil transakci, pokud není vyvolána žádná neošetřená výjimka.
- Silné pojmenování sestavení: Sestava musí být v mezipaměti GAC (Global Assembly Cache) pevně pojmenována a registrována. Sestava je registrována v COM + explicitně nebo opožděně zaregistrovaná po registraci v GAC.
- Registrace sestavení v COM +: Příprava sestavení pro vystavení klientům COM. Poté vytvořte knihovnu typů pomocí nástroje pro registraci sestavy, regasm.exe.

  ```
  regasm UnmanagedToManagedXa.dll
  ```

- Zaregistrujte sestavu na serveru GAC `gacutil /i UnmanagedToManagedXa.dll`.
- Zaregistrujte sestavení v modelu COM + pomocí instalačního nástroje služeb .NET, regsvcs.exe. Prohlédněte si knihovnu typu vytvořenou programem regasm.exe:

  ```
  Regsvcs /appname:UnmanagedToManagedXa  /tlb:UnmanagedToManagedXa.tlb
  UnmanagedToManagedXa.dll
  ```

- Sestava je implementována do aplikace GAC a později je registrována v COM + tím, že je zavedena opožděná registrace. Rámec .NET se stará o registraci po prvním spuštění kódu.

Příklad toku kódu pomocí modelu System.EnterpriseServices a System.Transactions s COM + jsou popsány v následujících sekcích:

Příklad toku kódu pomocí modelu System.EnterpriseServices

```
using System;
using IBM.WMQ;
using IBM.WMQ.Nmqi;
using System.Transactions;
using System.EnterpriseServices;

namespace UnmanagedToManagedXa
{

[ComVisible(true)]
[System.EnterpriseServices.Transaction(System.EnterpriseServices.TransactionOption.Required)
]
    public class MyXa : System.EnterpriseServices.ServicedComponent
    {

        public MQQueueManager QMGR = null;
        public MQQueueManager QMGR1 = null;
        public MQQueue QUEUE = null;
        public MQQueue QUEUE1 = null;
        public MQPutMessageOptions pmo = null;
        public MQMessage MSG = null;

        public MyXa()
        {
        }

        [System.EnterpriseServices.AutoComplete()]
        public void ExecuteUOW()
```

```
        {
            QMGR = new MQQueueManager("usemq");

            QUEUE = QMGR.AccessQueue("SYSTEM.DEFAULT.LOCAL.QUEUE",
                                     MQC.MQOO_INPUT_SHARED +
                                     MQC.MQOO_OUTPUT +
                                     MQC.MQOO_BROWSE);

            pmo = new MQPutMessageOptions();
            pmo.Options = MQC.MQPMO_SYNCPOINT;
            MSG = new MQMessage();
            QUEUE.Put(MSG, pmo);
            QMGR.Disconnect();
        }
    }
}


public void RunNow()
{
   MyXa xa = new MyXa();
   xa.ExecuteUOW();
}
```

Příklad toku kódu pomocí System.Transactions pro interakce s COM +

```
[STAThread]
public void ExecuteUOW()
{
Hashtable t1 = new Hashtable();
t1.Add(MQC.CHANNEL_PROPERTY, "SYSTEM.DEF.SVRCONN");
t1.Add(MQC.HOST_NAME_PROPERTY, "localhost");
t1.Add(MQC.PORT_PROPERTY, 1414);
t1.Add(MQC.TRANSPORT_PROPERTY, MQC.TRANSPORT_MQSERIES_CLIENT);
TransactionOptions opts = new TransactionOptions();

using(TransactionScope scope = new  TransactionScope(TransactionScopeOption.RequiresNew,
                                opts,     EnterpriseServicesInteropOption.Full)
  {

        QMGR = new MQQueueManager("usemq", t1);
        QUEUE = QMGR.AccessQueue("SYSTEM.DEFAULT.LOCAL.QUEUE",
                                          MQC.MQOO_INPUT_SHARED +
                                          MQC.MQOO_OUTPUT +
                                          MQC.MQOO_BROWSE);

         pmo = new MQPutMessageOptions();
         pmo.Options = MQC.MQPMO_SYNCPOINT;
         MSG = new MQMessage();
         QUEUE.Put(MSG, pmo);
         scope.Complete();
  }
 QMGR.Disconnect();
}
```

Vytváření jednoduchých vložení a získání zpráv v rámci TransactionScope

Ukázkové aplikace C# produktu jsou k dispozici v rámci produktu WebSphere MQ. Tyto jednoduché aplikace demonstrují vložení a získání zpráv v rámci TransactionScope. Na konci úlohy budete moci vkládat a získávat zprávy z fronty nebo tématu.

Než začnete

Služba MSDTC musí být spuštěna a povolena pro transakce XA.

Informace o této úloze

Příklad je jednoduchou aplikací, `SimpleXAPut` a `SimpleXAGet`. Programy `SimpleXAPut` a `SimpleXAGet` jsou C# aplikací, které jsou k dispozici v rámci produktu WebSphere MQ. Produkt `SimpleXAPut` demonstruje použití MQPUT v rámci distribuovaných transakcí pomocí oboru názvů SystemTransactions . Produkt `SimpleXAGet` demonstruje použití MQGET v části Distribuované transakce pomocí oboru názvů SystemTransactions .

`SimpleXAPut` se nachází v `WebSphere MQ\tools\dotnet\samples\cs\base`

Postup

Aplikace mohou být spuštěny s parametry příkazového řádku z produktu `tools\dotnet\samples\cs\base\bin`.

```
SimpleXAPut.exe -d destinationURI [-h host -p port -l channel -tx transaction -tm mode -n
numberOfMsgs]
```

```
SimpleXAGet.exe -d destinationURI [-h host -p port -l channel -tx transaction -tm mode -n
numberOfMsgs]
```

kde parametry jsou:

-destinationURI

Může jít o frontu nebo téma. Pro frontu zadejte jako `queue://queueName` a pro téma uveďte jako `topic://topicName`.

-host

Může se jednat o název hostitele, jako je například localhost nebo adresa IP.

-port

Port, na kterém je spuštěn správce front.

-channel

Kanál připojení, který se používá. Předvolba je `SYSTEM.DEF.SVRCONN`

-transaction

Výsledek transakce, například potvrzení nebo odvolání transakce.

-mode

Přenosový režim, například spravovaný nebo nespravovaný.

-numberOfMsgs

Počet zpráv. Výchozí hodnota je 1.

Příklad

```
SimpleXAPut -d topic://T01 -h localhost -p 2345 -tx rollback -tm unmanaged
```

```
SimpleXAGet -d queue://Q01 -h localhost -p 2345 -tx rollback -tm unmanaged
```

Obnova transakcí

Tento oddíl popisuje proces obnovení transakcí v produktu WebSphere MQ .NET XA pomocí spravovaného režimu.

Přehled

Ve zpracování distribuovaných transakcí mohou být transakce úspěšně dokončeny. Mohou však existovat scénáře, ve kterých může transakce selhat z řady důvodů. Mezi tyto důvody může patřit selhání systému, selhání hardwaru, chyba sítě, nesprávná nebo neplatná data, chyby aplikace nebo přírodní nebo člověkem způsobené pohromy. Selhání transakce není možné zabránit selháním transakce. Distribuovaný transakční systém musí být schopen zpracovat tato selhání. Musí být schopen detekovat a opravit chyby, když se vyskytnou. Tento proces je znám jako Zotavení transakcí.

Důležitým aspektem zpracování distribuovaných transakcí je obnova neúplných nebo sporných transakcí. Je nezbytné spustit zotavení, protože část pracovní transakce jednotky práce je zadržena, dokud nebude zotavena. Služba Microsoft .NET z knihovny tříd System.Transactions poskytuje volbu pro obnovení

neúplných/neověřených transakcí. Tato podpora zotavení očekává, že produkt Resource Manager udržuje protokoly transakcí a v případě potřeby spustí zotavení.

Model obnovení

V modelu zotavení transakce Microsoft .NET, správce transakcí (System.Transactionsnebo Microsoft Distributed Transaction coordinator (MS DTC) nebo obojí), iniciuje, koordinuje a řídí zotavení transakce. Správci prostředků založené na protokolu OLE Tx Protocol (Microsoft's XA protocol) poskytují volby pro konfiguraci DTC pro řízení, koordinaci a řízení obnovy pro ně. Aby to bylo možné provést, musí správci prostředků zaregistrovat XA_Switch s MS DTC pomocí nativního rozhraní.

Modul XA_Switch poskytuje vstupní body funkcí XA jako xa_start, xa_end a xa_recover ve správci Resource Manager k koordinátorovi distribuovaných transakcí.

Zotavení pomocí produktu Microsoft Distributed Transaction coordinator (DTC):

Koordinátor distribuovaných transakcí Microsoft poskytuje dva druhy procesů zotavení.

Studené zotavení

Studené zotavení se provádí v případě, že dojde k selhání procesu správce transakcí při otevření připojení ke správci prostředků XA. Po restartování správce transakcí načte protokoly správce transakcí a znovu zavede připojení ke správci prostředků XA a poté zahájí zotavení.

Zotavení po provozu

Obnova za provozu se provádí, pokud správce transakcí zůstane spuštěný, zatímco připojení mezi správcem transakcí a správcem prostředků XA selže, protože selže správce prostředků XA nebo síť. Po selhání se správce transakcí pravidelně pokouší znovu připojit ke správci prostředků XA. Po opětnému zavedení připojení zahájí správce transakcí zotavení XA.

Obor názvů System.Transactions poskytuje spravovanou implementaci distribuovaných transakcí, které jsou založeny na MS DTC jako správce transakcí. Poskytuje podobné funkce jako nativní rozhraní MS DTC, ale v plně spravovaném prostředí. Jediný rozdíl je o obnově transakce. System.Transactions očekává, že správci prostředků budou sami řídit obnovu a pak je koordinovat se správci transakcí (MS DTC). Resource Manager musí požádat o zotavení konkrétní nedokončené transakce a poté ji přijímá a koordinuje na základě skutečného výsledku dané transakce.

Proces zotavení transakce pro produkt WebSphere MQ .NET

Tento oddíl popisuje, jak mohou být distribuované transakce obnoveny pomocí tříd WebSphere MQ .NET.

Přehled

Chcete-li obnovit nedokončenou transakci, jsou požadovány informace o obnově. Informace o obnově transakcí musí být protokolovány do úložiště ze strany správců prostředků. Třídy WebSphere MQ .NET postupují podle podobné cesty. Informace o obnově transakcí jsou protokolovány do systémové fronty s názvem SYSTEM.DOTNET.XARECOVERY.QUEUE.

Zotavení transakcí v produktu WebSphere MQ .NET je dvoufázový proces.

1. Protokolování informací o obnově transakcí.
   - Pro každou transakci je během prepare fáze přidána trvalá zpráva obsahující informace o obnově do SYSTEM.DOTNET.XARECOVERY.QUEUE.
   - Pokud je potvrzení úspěšné, zpráva se odstraní.
2. Obnova transakcí pomocí monitorovací aplikace WmqDotnetXAMonitor.
   - WmqDotnetXAMonitor je spravovaná aplikace prostředí .NET, která zpracovává zprávy v systému SYSTEM.DOTNET.XARECOVERY.QUEUE a obnovení nedokončených transakcí

   Pokud program MCA nemůže vložit zprávu do cílové fronty, vygeneruje zprávu o výjimce obsahující původní zprávu a vloží ji do přenosové fronty, která má být odeslána do fronty pro odpovědi určené

v původní zprávě. (Je-li fronta pro odpovědi ve stejném správci front jako MCA, zpráva se umístí přímo do této fronty, nikoli do přenosové fronty.)

SYSTEM.DOTNET.XARECOVERY.QUEUE

Jedná se o systémovou frontu, která zadržuje informace o obnově transakcí nedokončených transakcí. Tato fronta se vytvoří, když se vytvoří správce front.

Poznámka: Neměli byste odstraňovat SYSTEM.DOTNET.XARECOVERY.QUEUE fronta.

Aplikace WMQDotnetXAMonitor

Produkt WebSphere MQ .NET XA Monitor monitoruje daného správce front a obnovuje nedokončené transakce, pokud existují. Následující text se považuje za nedokončené transakce a je obnoven:

Neúplné transakce

- Je-li transakce připravena, ale COMMIT nebyl dokončen během časového limitu.
- Je-li transakce připravena, ale správce front produktu WebSphere MQ nešel dolů.
- Je-li transakce připravena, ale pak se správce transakcí vypnul.

  Monitorovací aplikace musí být spuštěna ze stejného systému, kde je spuštěna aplikace klienta WebSphere MQ .NET. Existují-li aplikace spuštěné ve více systémech, které se připojují ke stejnému správci front, musí být aplikace monitoru spuštěna ze všech systémů. Ačkoli má každý klientský počítač spuštěnou aplikaci monitorování pro obnovení aplikace, každý monitor by měl být schopen identifikovat zprávu, která odpovídá transakci, kterou lokální monitorovací systém MS DTC koordinoval, aby mohl znovu uvést seznam a dokončit jej.

Případy použití zotavení transakce pro produkt WebSphere MQ .NET

Níže jsou uvedeny různé scénáře použití:

- **WebSphere MQ Application používající jednu DTC a jednu instanci správce front:** V tomto scénáři se při připojení ke správci front a spuštění transakce (UoW) v transakci a v případě selhání transakce a stane nekompletní, aplikace Monitor obnoví transakci a dokončí ji.

  V tomto scénáři bude spuštěna jediná instance aplikace monitorování, protože k transakcím je přidružen jediný správce front.
- **Několik aplikací produktu WebSphere MQ pomocí jediné instance DTC a jedné instance správce front:** V tomto scénáři existuje více než jedna aplikace WMQ pod jedním DTC a všechny se připojují ke stejnému správci front a spouštějí UoW v rámci transakcí.

  Pokud se transakce nezdaří a stanou se neúplnými, obnoví je aplikace Monitor a dokončí transakce související se všemi aplikacemi.

  V tomto scénáři se spustí jedna monitorovací aplikace, protože v transakcích se používá jeden správce front.
- **Více aplikací produktu WebSphere MQ , více DTC, různé instance správce front:** V tomto scénáři existuje více než jedna aplikace WMQ pod různými kódy DTC (tj. každá aplikace je spuštěna na jiném počítači) a připojuje se k různým správcům front.

  Pokud dojde k selhání a transakce se stane neúplnou, monitorovací aplikace zkontroluje TransactionManager, kde se nachází ve zprávě, aby určila adresu DTC. Pokud hodnota Whereabouts TransactionManagerodpovídá adrese DTC, pod kterou je monitor spuštěn, dokončí obnovu, jinak pokračuje v hledání, dokud nebude nalezena zpráva odpovídající jejímu DTC.

  V tomto scénáři bude existovat pouze jedna instance aplikace monitoru běžící na klienta (uživatel nebo počítač), protože každý klient má svého vlastního správce front použitého v transakcích.
- **Více aplikací produktu WebSphere MQ , více instancí DTC, více stejných instancí správců front:** V tomto scénáři existuje více než jedna aplikace WMQ pod různými kódy DTC (každá aplikace je spuštěna na jiném počítači) a všechny se připojují ke stejnému správci front.

Pokud dojde k selhání a transakce se stane neúplnou, monitorovací aplikace ověří TransactionManagerWhereoabouts ve zprávě, aby zkontroloval, zda adresa DTC a hodnota odpovídají DTC, pod kterým je monitor spuštěn. Pokud se obě hodnoty shodují, skončí obnova pokračuje v hledání, dokud nenajde zprávu odpovídající jejímu DTC.

V tomto scénáři bude existovat pouze jediná instance monitorování aplikace na klienta (uživatel nebo počítač), protože každý klient má své vlastní přidružení správce front použité v transakcích.

- **Několik aplikací WebSphere MQ Applications, single DTC, různé instance správce front:** V tomto scénáři existuje více než jedna aplikace WMQ pod jedním DTC (tj. na počítači, existuje více než jedna aplikace WMQ spuštěná) a připojuje se k různým správcům front.

  Pokud transakce selže a stane se neúplnou, monitorovací aplikace obnoví transakci.

  V tomto scénáři bude existovat tolik instancí aplikace monitorování, které jsou spuštěny jako správci front připojené, protože každá aplikace má svého vlastního správce front použitého v transakcích a každá z nich musí být zotavena.

  Poznámka: Není-li aplikace monitoru spuštěna na pozadí, můžete ji spustit.

Použití aplikace WMQDotnetXAMonitor

Aplikace monitoru XA musí být spuštěna ručně. Může být spuštěn kdykoliv. Můžete jej spustit, když se zobrazí zprávy v systému SYSTEM.DOTNET.XARECOVERY.QUEUE nebo ji můžete nechat běžet na pozadí před tím, než provedete libovolnou transakční práci s aplikacemi, které jsou zapsány pomocí tříd platformy .NET produktu WebSphere MQ .

Příkaz pro spuštění aplikace monitoru

```
WmqDotnetXAMonitor.exe -m <QueueManagerName> -n <ConnectionName> -c <Channel> -i
```

Kde

- **n** -Název připojení ve formátu hostitele (port). Název připojení může obsahovat více než jeden název připojení. Vícenásobné názvy připojení musí být uvedeny v seznamu s čárkami jako oddělovači, například "localhost (1414), localhost (1415), localhost (1416)". Monitor aplikace spustí obnovu pro každý z názvů připojení uvedených v seznamu odděleném čárkami.
- **c** -Název kanálu.
- **m** -Název správce front. Volitelné
- **i** -Dokončení heuristického větvení. Volitelné

Monitorovací aplikace provádí následující akce:

1. Kontroluje hloubku fronty SYSTEM.DOTNET.XARECOVERY.QUEUE v intervalu 100 sekund.
2. Je-li hloubka fronty větší než nula, monitor XA prochází frontu pro zprávy a kontroluje, zda zpráva odpovídá neúplným kritériím transakce.
3. Pokud některá ze zpráv odpovídá neúplným kritériím transakce, monitor jej stáhne a načte informace o zotavení transakce.
4. Poté určí, zda se informace o obnově vztahují k lokálnímu MS DTC. Pokud ano, pak bude pokračovat v obnově transakce. Jinak se vrátí k procházení další zprávy.
5. Pak zavolá správci front, aby obnovil nedokončenou transakci.

Nastavení konfiguračního souboru aplikace WmqDotNETXAMonitor

Chcete-li monitorovat aplikaci, lze vstupy poskytnout také pomocí konfiguračního souboru aplikace. Ukázkový konfigurační soubor aplikace je dodáván s produktem WebSphere MQ .NET. Tento soubor může být upraven podle vašich požadavků.

Konfigurační soubor aplikace má nejvyšší prioritu při zvažování vstupních hodnot. Pokud jsou vstupní hodnoty poskytnuty na příkazovém řádku i v konfiguračním souboru aplikace, jsou brány v úvahu i hodnoty z konfigurace aplikace.

Ukázkový konfigurační soubor aplikace.

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
<configSections>
<sectionGroup name="IBM.WMQ">
<section name="dnetxa" type="System.Configuration.NameValueFileSectionHandler" />
</sectionGroup>
</configSections>
<IBM.WMQ>
<dnetxa>
<add key="ConnectionName" value=""/>
<add key="ChannelName" value="" />
<add key="QueueManagerName" value="" />
<add key="UserId" value="" />
<add key="SecurityExit" value="" />
<add key="SecurityExitUserData" value = "">
</dnetxa>
</dnetxa>
</configuration>
```

WmqDotNetXAMonitor
Aplikace monitorování vytvoří soubor protokolu v adresáři aplikace pro protokolování průběhu monitorování a stavu zotavení transakce. Protokolování začíná názvem připojení a podrobnostmi kanálu, aby bylo možné zobrazit aktuálního správce front, pro kterého je zotavení spuštěno.

Jakmile je obnova spuštěna, MessageId zprávy o zotavení transakce, TransactionId nedokončené transakce a skutečný výsledek transakce jako transakce Transaction Manager Coordination bude protokolována.

Ukázkový soubor protokolu:

```
Time|ProcessId|ThreadId|WMQ .NET XA Recovery Monitor, Running now for
ConnectionName:xxxx, Time|ProcessId|ThreadId|Channel=xxxx
Time|ProcessId|ThreadId|Current QueueDepth = n
Time|ProcessId|ThreadId|Current MessageId = xxxx
Time|ProcessId|ThreadId|Current Incomplete Transaction being recovered = xxxxx
Time|ProcessId|ThreadId|Actual Outcome of the transaction(as per DTC)= Commit/Rollback
Time|ProcessId|ThreadId|Recovery Completed for TransactionId= xxxxx
Time|ProcessId|ThreadId|Current QueueDepth = n
Time|ProcessId|ThreadId|Current MessageId = xxxx
Time|ProcessId|ThreadId|Current Incomplete Transaction being recovered = xxxxx
Time|ProcessId|ThreadId|Actual Outcome of the transaction(as per DTC)= Commit/Rollback
Time|ProcessId|ThreadId| Recovery Completed for TransactionId= xxxxx
```

Použití tříd produktu WebSphere MQ pro prostředí .NET

Tato kolekce témat popisuje, jak nakonfigurovat systém ke spuštění ukázkových programů pro ověření instalace tříd WebSphere MQ pro instalaci .NET a o tom, jak spouštět vlastní programy.

Konfigurace správce front tak, aby přijímal klientská připojení TCP/IP

Chcete-li nakonfigurovat správce front tak, aby přijímal příchozí požadavky na připojení od klientů, postupujte takto:

1. Definujte kanál připojení serveru:
   a. Spusťte správce front.
   b. Definovat ukázkový kanál s názvem NET.CHANNEL[3]:

      ```
      DEF CHL('NET.CHANNEL') CHLTYPE(SVRCONN) TRPTYPE(TCP) MCAUSER(' ') +
      DESCR('Sample channel for WebSphere MQ classes for .NET')
      ```

2. Spustit modul listener:

   ```
   runmqlsr -t tcp [-m qmnqme] [-p portnum]
   ```

[3] V této ukázce neposuzujeme bezpečnostní důsledky. V případě produkčního systému zvažte použití zabezpečení SSL nebo ukončení zabezpečení. Další informace najdete v tématu Zabezpečení .

Poznámka: Hranaté závorky označují volitelné parametry; *qmname* není vyžadováno pro výchozího správce front a číslo portu *číslo portu* není povinné, pokud používáte výchozí hodnotu (1414).

Ukázkové aplikace

Chcete-li spustit vlastní aplikace .NET, použijte pokyny pro ověřovací programy a nahraďte místo ukázkových aplikací název vaší aplikace.

Dodává se pět ukázkových aplikací:

- Aplikace pro vkládání zpráv
- Aplikace pro získání zprávy
- Aplikace 'Ahoj světe'
- Aplikace typu publikování/odběr.
- Aplikace používající vlastnosti zprávy

Všechny tyto ukázkové aplikace jsou dodávány v jazyce C# a některé jsou také dodávány v jazycích C++ a Visual Basic. Aplikace můžete psát v libovolném jazyce, který podporuje prostředí .NET.

"Vložit zprávu" program SPUT (nmqsput.cs, mmqsput.cpp, vmqsput.vb)
Tento program ukazuje, jak vložit zprávu do pojmenované fronty. Program má tři parametry:

- Název fronty (povinné), například SYSTEM.DEFAULT.LOCAL.QUEUE
- Název správce front (volitelný)
- Definice kanálu (volitelné), například SYSTEM.DEF.SVRCONN/TCP/hostname(1414)

Není-li zadán žádný název správce front, správce front se standardně nastaví na výchozího lokálního správce front. Je-li kanál definován, má stejný formát jako proměnná prostředí MQSERVER.

"Get message" program SGET (nmqsget.cs, mmqsget.cpp, vmqsget.vb)
Tento program ukazuje, jak získat zprávu z pojmenované fronty. Program má tři parametry:

- Název fronty (povinné), například SYSTEM.DEFAULT.LOCAL.QUEUE
- Název správce front (volitelný)
- Definice kanálu (volitelné), například SYSTEM.DEF.SVRCONN/TCP/hostname(1414)

Není-li zadán žádný název správce front, správce front se standardně nastaví na výchozího lokálního správce front. Je-li kanál definován, má stejný formát jako proměnná prostředí MQSERVER.

Program "Ahoj světe" (nmqwrld.cs, mmqwrld.cpp, vmqwrld.vb)
Tento program ukazuje, jak vložit a získat zprávu. Program má tři parametry:

- Název fronty (volitelné), například SYSTEM.DEFAULT.LOCAL.QUEUE nebo SYSTEM.DEFAULT.MODEL.QUEUE
- Název správce front (volitelný)
- Definice kanálu (volitelná), například SYSTEM.DEF.SVRCONN/TCP/hostname(1414)

Není-li zadán žádný název fronty, bude použit výchozí název SYSTEM.DEFAULT.LOCAL.QUEUE. Není-li zadán žádný název správce front, správce front se standardně nastaví na výchozího lokálního správce front.

Program "Publish/subscribe" (MQPubSubSample.cs)
Tento program obsahuje informace o způsobu použití publikování a odběru produktu WebSphere MQ . Dodává se pouze v C#. Program má dva parametry:

- Název správce front (volitelný)
- Definice kanálu (volitelná).

Program vlastností "Message properties" (MQMessagePropertiesSample.cs)
Tento program ukazuje, jak používat vlastnosti zpráv. Dodává se pouze v C#. Program má dva parametry:

- Název správce front (volitelný)

- Definice kanálu (volitelná).

Vaši instalaci můžete ověřit kompilací a spuštěním těchto aplikací.

Ukázkové aplikace jsou instalovány do následujících umístění podle jazyka, v němž jsou napsány. *MQ_INSTALLATION_PATH* Představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

C#

MQ_INSTALLATION_PATH\Tools\dotnet\samples\cs\nmqswrld.cs

MQ_INSTALLATION_PATH\Tools\dotnet\samples\cs\nmqsput.cs

MQ_INSTALLATION_PATH\Tools\dotnet\samples\cs\nmqsget.cs

MQ_INSTALLATION_PATH\Tools\dotnet\samples\cs\MQPubSubSample.cs

MQ_INSTALLATION_PATH\Tools\dotnet\samples\cs\MQMessagePropertiesSample.cs

Managed C++

MQ_INSTALLATION_PATH\Tools\dotnet\samples\mcp\mmqswrld.cpp

MQ_INSTALLATION_PATH\Tools\dotnet\samples\mcp\mmqsput.cpp

MQ_INSTALLATION_PATH\Tools\dotnet\samples\mcp\mmqsget.cpp

Visual Basic

MQ_INSTALLATION_PATH\Tools\dotnet\samples\vb\vmqswrld.vb

MQ_INSTALLATION_PATH\Tools\dotnet\samples\vb\vmqsput.vb

MQ_INSTALLATION_PATH\Tools\dotnet\samples\vb\vmqsget.vb

MQ_INSTALLATION_PATH\Tools\dotnet\samples\vb\xmqswrld.vb

MQ_INSTALLATION_PATH\Tools\dotnet\samples\vb\xmqsput.vb

MQ_INSTALLATION_PATH\Tools\dotnet\samples\vb\xmqsget.vb

Chcete-li sestavit ukázkové aplikace, byl pro každý jazyk dodán dávkový soubor.

C#

MQ_INSTALLATION_PATH\Tools\dotnet\samples\cs\bldcssamp.bat

Soubor bldcssamp.bat obsahuje řádek pro každý vzorek, což je vše, co je nezbytné k sestavení tohoto ukázkového programu:

```
csc /t:exe /r:System.dll /r:amqmdnet.dll /lib:MQ_INSTALLATION_PATH\bin
/out:nmqwrld.exe nmqwrld.cs
```

Managed C++

MQ_INSTALLATION_PATH\Tools\dotnet\samples\mcp\bldmcpsamp.bat

Soubor bldmcpsamp.bat obsahuje řádek pro každý vzorek, což je vše, co je nezbytné k sestavení tohoto ukázkového programu:

```
cl /clr:oldsyntax MQ_INSTALLATION_PATH\bin mmqwrld.cpp
```

Chcete-li tyto aplikace zkompilovat v produktu Microsoft Visual Studio 2003/.NET SDKv1.1, nahraďte kompilační příkaz:

```
cl /clr:oldsyntax MQ_INSTALLATION_PATH\bin mmqwrld.cpp
```

s

```
cl /clr MQ_INSTALLATION_PATH\bin mmqwrld.cpp
```

Visual Basic

MQ_INSTALLATION_PATH\Tools\dotnet\samples\vb\bldvbsamp.bat

Soubor bldvbsamp.bat obsahuje řádek pro každý vzorek, což je vše, co je nezbytné k sestavení tohoto ukázkového programu:

```
vbc /r:System.dll /r:MQ_INSTALLATION_PATH\bin\amqmdnet.dll /out:vmqwrld.exe vmqwrld.vb
```

Řešení problémů s produktem WebSphere MQ .NET

Pokud se program nedokončí úspěšně, spusťte jednu z ukázkových aplikací a postupujte podle doporučení uvedených v diagnostických zprávách.

Tyto ukázkové aplikace jsou popsány v tématu “Použití tříd produktu WebSphere MQ pro prostředí .NET” na stránce 552.

Pokud problémy přetrvávají a vy potřebujete kontaktovat servisní tým IBM , můžete být požádáni o zapnutí trasovacího prostředku.

Trasování ukázkové aplikace

Pokyny pro použití trasovacího prostředku naleznete v části “Trasování programů WebSphere MQ .NET” na stránce 574.

Chybové zprávy

Může se zobrazit následující společná chybová zpráva:

Neošetřená výjimka typu 'System.IO.FileNotFoundException' došlo v neznámém

Pokud k této chybě dojde buď pro amqmdnet.dll nebo amqmdxcs.dll, buď zajistěte, aby obě byly registrovány v mezipaměti 'Global Assembly Cache', nebo vytvořte konfigurační soubor, který ukazuje na montážní celky amqmdnet.dll a amqmdxcs.dll . Obsah mezipaměti sestavení můžete zkontrolovat a změnit pomocí souboru mscorcfg.msc, který je dodáván jako součást rámce .NET.

Pokud nebyl rámec .NET k dispozici při instalaci produktu WebSphere MQ , nemusí být tyto třídy registrovány v globální mezipaměti sestavení. Proces registrace můžete ručně znovu spustit pomocí příkazu

```
amqidnet -c MQ_INSTALLATION_PATH\bin\amqidotn.txt -l logfile.txt
```

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

Informace o této instalaci se zapíší do uvedeného souboru protokolu (`logfile.txt` v tomto příkladu).

Psaní a implementace programů WebSphere MQ .NET

Chcete-li používat třídy WebSphere MQ pro .NET pro přístup k frontám produktu WebSphere MQ , můžete psát programy v libovolném jazyce, který je podporován rozhraním .NET, který obsahuje zprávy pro vkládání zpráv do front zpráv WebSphere MQ a jejich získání od nich.

Dokumentace k produktu WebSphere MQ obsahuje informace pouze v jazycích C#, C++ a Visual Basic.

Tato kolekce témat obsahuje informace, které vám pomohou při psaní aplikací pro interakci se systémy WebSphere MQ . Podrobnosti o jednotlivých třídách naleznete v tématu Třídy a rozhraní produktu WebSphere MQ .NET.

Rozdíly spojení

Způsob, jakým program pro produkt WebSphere MQ .NET obsahuje některé závislosti na režimech připojení, které chcete použít.

Připojení spravovaného klienta

Jsou-li třídy WebSphere MQ pro .NET používány jako spravovaného klienta, existuje řada rozdílů ze standardního klienta WebSphere MQ MQI.

Pro spravovaného klienta nejsou k dispozici následující funkce:

- Komprese kanálu
- Podpora SSL
- Řetězení uživatelských procedur kanálu

Pokusíte-li se tyto funkce používat se spravovaným klientem, bude vrácena výjimka MQException. Je-li chyba zjištěna na straně klienta připojení, použije kód příčiny MQRC_ENVIRONMENT_ERROR. Je-li zjištěn na konci serveru, použije se kód příčiny vrácený serverem.

Uživatelské procedury kanálu zapsané pro nespravovaného klienta nepracují. Musíte zapsat nové uživatelské procedury speciálně pro spravovaného klienta. Zkontrolujte, že v tabulce definic kanálů klienta (CCDT) nejsou zadány žádné platné uživatelské procedury kanálu.

Název uživatelské procedury spravovaného kanálu může mít délku až 999 znaků. Pokud však k určení názvu uživatelské procedury kanálu použijete tabulky CCDT, je tato hodnota omezena na 128 znaků.

Komunikace je podporována pouze přes TCP/IP.

Pokud zastavíte správce front pomocí příkazu **endmqm** , může kanál připojení serveru ke spravovanému klientu .NET trvat delší dobu než kanál připojení serveru k jiným klientům.

Pokud jste nastavili proměnnou *NMQ_MQ_LIB* na hodnotu `managed` , chcete-li používat diagnostiku problémů spravovaného produktem WebSphere MQ , není podporován žádný z parametrů -i, -p, -s, -b nebo -c příkazu **strmqtrc** .

Spravovaná aplikace .NET, která používá transakce XA, nebude pracovat se správcem front z/OS . Spravovaný. Síťový klient, který se pokouší připojit ke správci front z/OS , selže s chybou MQRC_UOW_ENLISTMENT_ERROR (mqrc=2354), při volání MQOPEN. Spravovaná aplikace .NET, která používá transakce XA, však bude pracovat s distribuovaným správcem front.

Definování, který typ připojení se má použít

Typ připojení je určen nastavením názvu připojení, názvu kanálu, hodnoty přizpůsobení NMQ_MQ_LIB a vlastností MQC.TRANSPORT_PROPERTY.

Název připojení můžete zadat takto:

- Explicitně na konstruktoru MQQueueManager :

```
public MQQueueManager(String queueManagerName, MQLONG Options, string Channel,
string ConnName)
```

```
public MQQueueManager(String queueManagerName, string Channel, string ConnName)
```

- Nastavením vlastností MQC.HOST_NAME_PROPERTY a, volitelně, MQC.PORT_PROPERTY v položce hašovací tabulky v konstruktoru MQQueueManager :

```
public MQQueueManager(String queueManagerName, Hashtable properties)
```

- Jako explicitní hodnoty MQEnvironment

```
MQEnvironment.Hostname
```

`MQEnvironment.Port`(volitelné).

- Nastavením vlastností MQC.HOST_NAME_PROPERTY a, volitelně, MQC.PORT_PROPERTY v hašovací tabulce MQEnvironment.properties .

Název kanálu můžete zadat takto:

- Explicitně na konstruktoru MQQueueManager :

```
public MQQueueManager(String queueManagerName, MQLONG Options, string Channel,
string ConnName)
```

```
public MQQueueManager(String queueManagerName, string Channel, string ConnName)
```

- Nastavením vlastnosti MQC.CHANNEL_PROPERTY se nachází v položce hašovací tabulky v konstruktoru MQQueueManager :

```
public MQQueueManager(String queueManagerName, Hashtable properties)
```

- Jako explicitní hodnota MQEnvironment

```
MQEnvironment.Channel
```

- Nastavením vlastnosti MQC.CHANNEL_PROPERTY v hašovací tabulce MQEnvironment.properties .

Vlastnost přenosu můžete určit tímto způsobem:

- Nastavením vlastnosti MQC.TRANSPORT_PROPERTY v rámci položky hašovací tabulky v konstruktoru MQQueueManager :

```
public MQQueueManager(String queueManagerName, Hashtable properties)
```

- Nastavením vlastnosti MQC.TRANSPORT_PROPERTY v hašovací tabulce MQEnvironment.properties .

Vyberte typ připojení, který požadujete, pomocí jedné z následujících hodnot:

MQC.TRANSPORT_MQSERIES_BINDINGS -připojit se jako server
MQC.TRANSPORT_MQSERIES_CLIENT -připojit se jako klient bez standardu XA
MQC.TRANSPORT_MQSERIES_XACLIENT -připojit se jako klient XA
MQC.TRANSPORT_MQSERIES_MANAGED -připojit se jako spravovaný klient mimo XA

Hodnotu přizpůsobení NMQ_MQ_LIB můžete nastavit tak, aby explicitně vybíral typ připojení, jak ukazuje následující tabulka.

Hodnota NMQ_MQ_LIB	**Typ připojení**
mqic.dll	Připojit jako klient bez podpory XA
mqicxa.dll	Připojit jako klient XA
mqm.dll	Připojit jako server nebo jako klienta bez podpory XA
Spravováno	Připojit jako spravovaný klient bez podpory XA
Poznámka: Hodnoty proměnné mqic32.dll a mqic32xa.dll jsou při kompatibilitě se staršími verzemi přijímány jako synonyma souborů mqic.dll a mqicxa.dll . Avšak soubor mqm.dll a mqm.pdb jsou součástí balíku klienta pouze od verze 7.1 .	

Vyberete-li typ připojení, který není ve vašem prostředí k dispozici, například zadejte příkaz mqic32xa.dll a nemáte podporu standardu XA, produkt WebSphere MQ .NET vygeneruje výjimku.

Nastavení hodnoty NMQ_MQ_LIB na hodnotu "managed" způsobí, že klient bude používat diagnostické testy problémů produktu WebSphere MQ , převod dat .NET a další funkce WebSphere MQ na nižší úrovni spravovaného systému.

Všechny ostatní hodnoty parametru NMQ_MQ_LIB způsobí, že proces .NET bude používat nespravované diagnostické testy a převod dat produktu WebSphere MQ a další funkce WebSphere produktu MQ v nespravovaném systému (za předpokladu, že je v systému nainstalován klient nebo server WebSphere MQ MQI).

Produkt WebSphere MQ .NET volí typ připojení následujícím způsobem:

1. Pokud je MQC.TRANSPORT_PROPERTY je připojen v souladu s hodnotou proměnné MQC.TRANSPORT_PROPERTY.

   Všimněte si však, že nastavení MQC.TRANSPORT_PROPERTY na MQC.TRANSPORT_MQSERIES_MANAGED nezaručuje, že se proces klienta spustí. I s tímto nastavením není klient spravován v následujících případech:

   - Pokud jiný podproces v procesu navázalo spojení s MQC.TRANSPORT_PROPERTY nastaveno na něco jiného než MQC.TRANSPORT_MQSERIES_MANAGED.
   - Pokud není volba NMQ_MQ_LIB nastavena na hodnotu "managed", diagnostické testy problémů, konverze dat a další funkce nízké úrovně nejsou zcela spravovány (za předpokladu, že je na systému nainstalován klient nebo server WebSphere MQ MQI).
2. Pokud byl zadán název připojení bez názvu kanálu nebo byl-li zadán název kanálu bez názvu připojení, bude vrácena chyba.
3. Je-li zadán název připojení i název kanálu, postupujte takto:
   - Je-li hodnota NMQ_MQ_LIB nastavena na hodnotu mqic32xa.dll, připojí se jako klient XA.
   - Je-li hodnota NMQ_MQ_LIB nastavena na spravovaná, připojí se jako spravovaný klient.
   - Jinak se připojí jako klient bez podpory XA.
4. Je-li zadán parametr NMQ_MQ_LIB, připojí se k němu podle hodnoty NMQ_MQ_LIB.
5. Je-li server WebSphere MQ nainstalován, připojí se jako server.
6. Je-li nainstalován klient WebSphere MQ MQI, připojí se jako klient mimo XA.
7. Jinak se připojí jako spravovaný klient.

Konfigurační soubory pro třídy produktu WebSphere MQ pro prostředí .NET

Aplikace klienta prostředí .NET může používat konfigurační soubor klienta WebSphere MQ MQI a v případě, že používáte typ spravovaného připojení, konfigurační soubor aplikace .NET. Nastavení v konfiguračním souboru aplikace má prioritu.

Konfigurační soubor klienta

Aplikace klienta WebSphere MQ pro klientskou aplikaci prostředí .NET může používat konfigurační soubor klienta stejným způsobem jako jiný klient WebSphere MQ MQI. Tento soubor se obvykle nazývá mqclient.ini, ale můžete zadat jiný název souboru. Další informace o konfiguračním souboru klienta naleznete v tématu Konfigurace klienta pomocí konfiguračního souboru WebSphere MQ MQI client configuration file.

Pouze následující atributy v konfiguračním souboru klienta WebSphere MQ MQI jsou relevantní pro třídy produktu WebSphere MQ pro prostředí .NET. Určíte-li jiné atributy, nemá žádný efekt.

sekce	**Atribut**
kanály	CCSID
kanály	Adresář ChannelDefinition
kanály	ChannelDefinition

sekce	Atribut
kanály	Parametry ServerConnectionParms
ClientExit-cesta	ExitsDefaultPath
ClientExit-cesta	ExitsDefaultPath64
MessageBuffer	MaximumSize
MessageBuffer	PurgeTime
MessageBuffer	UpdatePercentage
TCP	ClntRcvBufSize
TCP	ClntSndBufSize
TCP	IPAddressVersion
TCP	KeepAlive

Všechny tyto atributy můžete potlačit pomocí příslušné proměnné prostředí.

Konfigurační soubor aplikace

Pracujete-li se spravovaným typem připojení, můžete také přepsat konfigurační soubor klienta WebSphere MQ a ekvivalentní proměnné prostředí pomocí konfiguračního souboru aplikace .NET.

Nastavení konfiguračního souboru aplikace .NET se postupuje pouze při spuštění s typem spravovaného připojení a jsou ignorovány pro jiné typy připojení.

Konfigurační soubor aplikace .NET a jeho formát jsou definovány společností Microsoft pro obecné použití v rámci .NET framework, ale určité názvy sekcí, klíče a hodnoty uvedené v této dokumentaci jsou specifické pro produkt Websphere MQ.

Formát konfiguračního souboru aplikace .NET je číslo *oddílů*. Každá sekce obsahuje jeden nebo více *klíčů*a každý klíč má přidruženou *hodnotu*. Následující příklad zobrazuje sekce, klíče a hodnoty použité v konfiguračním souboru aplikace .NET k řízení vlastnosti TCP/IP KeepAlive :

```
<configuration>
  <configSections>
    <section name="TCP" type="System.Configuration.NameValueSectionHandler"/>
  </configSections>
  <TCP>
    <add key="KeepAlive" value="true"></add>
  </TCP>
<configuration>
```

Klíčová slova použitá v názvech sekcí konfiguračního souboru aplikace .NET a klíčů se přesně shodují s klíčovými slovy pro Stanzy a atributy definované v konfiguračním souboru klienta.

Další informace naleznete v dokumentaci společnosti Microsoft .

Fragment kódu příkladu

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 a WebSphere MQ message, writing some text in UTF format
      MQMessage hello_world = new MQMessage();
      hello_world.WriteUTF("Hello World!");

      // Specify the message options
      MQPutMessageOptions pmo = new MQPutMessageOptions(); // accept the defaults,
```

```
                                                          // same as MQPMO_DEFAULT

      // Put the message on the queue
      system_default_local_queue.Put(hello_world, pmo);



      // Get the message back again

      // First define a WebSphere MQ message buffer to receive the message
      MQMessage retrievedMessage =new MQMessage();
      retrievedMessage.MessageId =hello_world.MessageId;

      // Set the get message options
      MQGetMessageOptions gmo =new MQGetMessageOptions(); //accept the defaults
                                                          //same as MQGMO_DEFAULT

      // Get the message off the queue
      system_default_local_queue.Get(retrievedMessage,gmo);

      // Prove we have the message by displaying the UTF message text
      String msgText = retrievedMessage.ReadUTF();
      Console.WriteLine("The message is: {0}", msgText);

      // Close the queue
      system_default_local_queue.Close();

      // Disconnect from the queue manager
      qMgr.Disconnect();
    }

    //If an error has occurred,try to identify what went wrong.

    //Was it a WebSphere MQ error?
    catch (MQException ex)
    {
      Console.WriteLine("A WebSphere MQ error occurred: {0}", ex.ToString());
    }

    catch (System.Exception ex)
    {
      Console.WriteLine("A System error occurred: {0}", ex.ToString());
    }

    return 0;
  }//end of start
}//end of sample
```

Operace pro správce front

Tento oddíl popisuje způsob připojení a odpojení od správce front s použitím tříd produktu WebSphere MQ pro prostředí .NET.

Nastavení prostředí produktu WebSphere MQ

Než použijete připojení klienta k připojení ke správci front, je třeba nastavit prostředí produktu WebSphere MQ .

Poznámka: Tento krok není nutný při použití produktu WebSphere MQ tříd pro prostředí vazeb serveru .NET v režimu .NET.

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, jako například v následujícím seznamu:

- Název kanálu
- Název hostitele
- Číslo portu
- Uživatelské procedury kanálu
- Parametry zabezpečení SSL
- ID uživatele a heslo

Chcete-li získat úplné informace o třídě MQEnvironment, prohlédněte si téma MQEnvironment Třída .NET

Chcete-li určit název kanálu a název hostitele, použijte následující kód:

```
MQEnvironment.Hostname = "host.domain.com";
MQEnvironment.Channel  = "client.channel";
```

Při výchozím nastavení se klienti pokusí připojit k modulu listener produktu WebSphere MQ na portu 1414. Chcete-li určit jiný port, použijte tento kód:

```
MQEnvironment.Port = nnnn;
```

Připojování ke správci front

Nyní jste připraveni připojit se ke správci front vytvořením nové instance třídy MQQueueManager :

```
MQQueueManager queueManager = new MQQueueManager("qMgrName");
```

Chcete-li se odpojit od správce front, volejte metodu Disconnect ve správci front:

```
queueManager.Disconnect();
```

Chcete-li se připojit ke správci front, musíte mít při pokusu o připojení ke správci front oprávnění k dotazům (inq). Bez dotazovacího oprávnění se pokus o připojení nezdaří.

Vočíte-li metodu Disconnect , všechny otevřené fronty a procesy, ke kterým jste přistoupil prostřednictvím tohoto správce front, budou zavřeny. Je však dobrým programovacím postupem, abyste tyto prostředky při jejich používání explicitně uzavřeli. Chcete-li zavřít prostředky, použijte metodu Close na objektu přidruženém ke každému prostředku.

Metody Commit a Backout ve správci front nahradí volání MQCMIT a MQBACK, která se používají spolu s procedurálním rozhraním.

Přístup k frontám a tématům

K frontám a tématům můžete přistupovat pomocí metod produktu MQQueueManager nebo příslušných konstruktorů.

Chcete-li přistupovat k frontám, použijte metody třídy MQQueueManager . MQOD (struktura deskriptoru objektu) je sbalena do parametrů těchto metod. Chcete-li například otevřít frontu ve správci front představovaném objektem MQQueueManager s názvem queueManager, použijte následující kód:

```
MQQueue queue = queueManager.AccessQueue("qName",
                                         MQC.MQOO_OUTPUT,
                                         "qMgrName",
                                         "dynamicQName",
                                         "altUserId");
```

Parametr *options* je stejný jako parametr Options v rámci volání MQOPEN.

Metoda AccessQueue vrací nový objekt třídy MQQueue.

Až skončíte s používáním fronty, použijte metodu Close () k zavření, jako v následujícím příkladu:

```
queue.Close();
```

S produktem WebSphere MQ .NET můžete vytvořit také frontu pomocí konstruktoru MQQueue. Parametry jsou přesně stejné jako u metody accessQueue spolu s přidáním parametru správce front, který určuje instanci objektu MQQueueManager , jež má být použita. Příklad:

```
MQQueue queue = new MQQueue(queueManager,
                            "qName",
                            MQC.MQOO_OUTPUT,
                            "qMgrName",
                            "dynamicQName",
                            "altUserId");
```

Při vytváření objektu fronty tímto způsobem lze zapisovat do vlastních podtříd fronty MQQueue.

Podobně můžete také přistupovat k tématům s použitím metod třídy MQQueueManager . Chcete-li otevřít téma, použijte metodu AccessTopic(). Tento příkaz vrátí nový objekt třídy MQTopic. Po dokončení práce s tématem použijte metodu Close () objektu MQTopic k jeho zavření.

Také můžete vytvořit téma pomocí konstruktoru MQTopic. Pro témata existuje řada konstruktorů; další informace viz téma MQTopic Třída .NET.

Práce se zprávami

Zprávy se zpracovávají pomocí metod fronty nebo tříd témat. Chcete-li vytvořit novou zprávu, vytvořte nový objekt MQMessageobject.

Vložení zpráv do front nebo témat pomocí metody Put () třídy MQQueue nebo MQTopic. Načtení zpráv z front nebo témat pomocí metody Get () třídy MQQueue nebo MQTopic. Na rozdíl od procedurálního rozhraní, kde příkazy MQPUT a MQGET umístili a získaní pole bajtů, třídy WebSphere MQ pro prostředí .NET put a get instance třídy MQMessage. Třída MQMessage zapouzdřuje vyrovnávací paměť dat, která obsahuje skutečná data zprávy, spolu se všemi parametry MQMD (Message Descriptor), které popisují danou zprávu.

Chcete-li vytvořit novou zprávu, vytvořte novou instanci třídy MQMessage a pomocí metod WriteXXX umístěte data do vyrovnávací paměti zpráv.

Když se vytvoří nová instance zprávy, všechny parametry MQMD se automaticky nastaví na jejich výchozí hodnoty, jak je definováno v Počáteční hodnoty a deklarace jazyka pro MQMD . Metoda PUT () fronty MQQueue také vezme instanci třídy voleb MQPutMessagejako parametr. Tato třída představuje strukturu MQPMO. Následující příklad vytvoří zprávu a vloží ji do fronty:

```
// Build a new message containing my age followed by my name
MQMessage myMessage = new MQMessage();
myMessage.WriteInt(25);

String name = "Charlie Jordan";
myMessage.WriteUTF(name);

// Use the default put message options...
MQPutMessageOptions pmo = new MQPutMessageOptions();

// put the message!
queue.Put(myMessage,pmo);
```

Metoda Get () MQQueue vrací novou instanci MQMessage, která představuje zprávu právě převzaté z fronty. Jako parametr má také instanci třídy voleb MQGetMessage. Tato třída představuje strukturu MQGMO.

Maximální velikost zprávy není třeba zadávat, protože metoda Get () automaticky upravuje velikost vnitřní vyrovnávací paměti tak, aby odpovídala příchozí zprávě. Použijte metody ReadXXX třídy MQMessage pro přístup k datům ve vrácené zprávě.

Následující příklad uvádí, jak získat zprávu z fronty:

```
// Get a message from the queue
MQMessage theMessage    = new MQMessage();
MQGetMessageOptions gmo = new MQGetMessageOptions();
```

```
queue.Get(theMessage,gmo);  // has default values

// Extract the message data
int age = theMessage.ReadInt();
String name1 = theMessage.ReadUTF();
```

Formát čísla, který metody čtení a zápisu používá, můžete změnit nastavením proměnné člena *encoding* .

Můžete změnit znakovou sadu, která má být použita pro čtení a zápis řetězců, a to nastavením proměnné člena *characterSet* .

Další podrobnosti viz MQMessage Třída .NET .

Poznámka: Metoda WriteUTF() MQMessage automaticky kóduje délku řetězce stejně jako bajty Unicode, které obsahuje. Když vaše zpráva bude čtena jiným programem .NET (pomocí ReadUTF()), je to nejjednodušší způsob, jak odeslat informace o řetězci.

Práce s vlastnostmi zprávy

Vlastnosti zpráv umožňují vybírat zprávy nebo načítat informace o zprávě bez přístupu k jejím záhlavím. Třída MQMessage obsahuje metody pro získání a nastavení vlastností.

Vlastnosti zpráv můžete použít k povolení výběru zpráv pro zpracování nebo načítání informací o zprávě bez přístupu k záhlavím MQMD nebo MQRFH2 . Usnadňují také komunikaci mezi aplikacemi WebSphere MQ a aplikací JMS. Další informace o vlastnostech zprávy v produktu WebSphere MQnaleznete v tématu Vlastnosti zprávy.

Třída MQMessage poskytuje řadu metod pro získání a nastavení vlastností v závislosti na typu dat vlastnosti. Metody get mají názvy formátu Get * Property a metody set mají názvy z formátu Set * Property, kde hvězdička (*) představuje jeden z následujících řetězců:

- Logická hodnota
- bajt
- Bajtů
- Dvojitá čára
- Pohyblivá desetinná čárka
- Int
- Int2
- Int4
- Int8
- Dlouhý
- Objekt
- Krátký
- Řetězec

Chcete-li například získat vlastnost myproperty produktu WebSphere MQ (znakový řetězec), použijte volání `message.GetStringProperty('myproperty')`. Volitelně můžete předat deskriptor vlastnosti, který bude produkt WebSphere MQ dokončen.

Ošetření chyb

Obsluhovat chyby ze tříd WebSphere MQ pro .NET za použití bloků `try` a `catch` .

Metody v rozhraní .NET nevrací kód dokončení a kód příčiny. Místo toho vygenerují výjimku vždy, když kód dokončení a kód příčiny pocházející z volání produktu WebSphere MQ nejsou obě nula. Tím se zjednodušíte logiku programu, takže nemusíte vracet návratové kódy po každém volání do produktu WebSphere MQ. Můžete se rozhodnout, ve kterých bodech ve vašem programu se chcete vypořádat

s možností selhání. V těchto bodech můžete obklopovat kód pomocí bloků `try` a `catch` , jako v následujícím příkladu:

```
try
{
  myQueue.Put(messageA,PutMessageOptionsA);
  myQueue.Put(messageB,PutMessageOptionsB);
}
catch (MQException ex)
{
  // This block of code is only executed if one of
  // the two put methods gave rise to a non-zero
  // completion code or reason code.
  Console.WriteLine("An error occurred during the put operation:" +
                        "CC = " + ex.CompletionCode +
                        "RC = " + ex.ReasonCode);
  Console.WriteLine("Cause exception:" + ex );
}
```

Získání a nastavení hodnot atributu

Třídy MQManagedObject, MQQueue a MQQueueManager obsahují metody, které vám umožňují získat a nastavit hodnoty atributu. Všimněte si, že metody pro frontu MQQueue pracují pouze v případě, že při otevření fronty zadáte příslušné parametry dotazu a nastavení.

Pro obecné atributy dědí třídy MQQueueManager a MQQueue ze třídy s názvem MQManagedObject. Tato třída definuje rozhraní Inquire () a Set ().

Když vytváříte nový objekt správce front pomocí operátoru *nový* , je automaticky otevřen pro zjištění. Použijete-li metodu AccessQueue() pro přístup k objektu fronty, tento objekt *není* automaticky otevřen pro dotazování nebo nastavení operací, což může způsobit problémy s některými typy vzdálených front. Chcete-li použít metody Inquire and Set a nastavit vlastnosti ve frontě, je třeba v parametru openOptions metody AccessQueue() určit příslušné parametry dotazu a nastavit příslušné parametry.

Metody dotazování a nastavení mají tři parametry:

- pole selektorů
- Pole intAttrs
- pole charAttrs

Nepotřebujete parametry SelectorCount, IntAttrCount a CharAttrLength, které se nacházejí v MQINQ, protože délka pole je vždy známá. Následující příklad uvádí, jak provést dotaz na frontu:

```
//inquire on a queue
int [ ] selectors = new int [2] ;
int [ ] intAttrs = new int [1] ;
byte [ ] charAttrs = new byte [MQC.MQ_Q_DESC_LENGTH];
selectors [0] = MQC.MQIA_DEF_PRIORITY;
selectors [1] = MQC.MQCA_Q_DESC;
queue.Inquire(selectors,intAttrs,charAttrs);
ASCIIEncoding enc = new ASCIIEncoding();
String s1 = "";
s1 = enc.GetString(charAttrs);
```

Všechny atributy těchto objektů mohou být dotazovány. Podmnožina atributů je vystavena jako vlastnosti objektu. Seznam atributů objektů najdete v tématu Atributy objektů. Pro vlastnosti objektu si prohlédněte odpovídající popis třídy.

Vícevláknové programy

Běhové prostředí .NET je z podstaty vícevláknové. Třídy produktu WebSphere MQ pro prostředí .NET umožňují sdílení objektu správce front mezi více podprocesy, ale zajišťuje synchronizaci všech přístupů k cílovému správci front.

Zvažte jednoduchý program, který se připojí ke správci front a otevře frontu při spuštění. Program zobrazí na obrazovce jediné tlačítko. Když uživatel klepne na toto tlačítko, program načte zprávu z fronty. V této

situaci se inicializace aplikace vyskytuje v jednom podprocesu a kód, který se provádí v odpovědi na stisknutí tlačítka, se provádí v samostatném podprocesu (podproces uživatelského rozhraní).

Implementace produktu WebSphere MQ .NET zajišťuje, že pro určité připojení (instance objektuMQQueueManager) bude veškerý přístup k cílovému správci front produktu WebSphere MQ synchronizován. Výchozí chování je, že podproces, který chce vydat volání správci front, je blokován, dokud nejsou dokončena všechna ostatní volání v průběhu tohoto připojení. Pokud vyžadujete souběžný přístup ke stejnému správci front z více podprocesů ve svém programu, vytvořte nový objekt MQQueueManager pro každý podproces, který vyžaduje souběžný přístup. (Toto je ekvivalent k zadání samostatného volání MQCONN pro každé vlákno.)

Pokud jsou výchozí volby připojení přepsány MQC.MQCNO_HANDLE_SHARE_NONE nebo MQC.MQCNO_SHARE_NO_BLOCK , poté správce front již není synchronizován.

Použití definiční tabulky kanálu klienta s rozhraním .NET

Pro produkt WebSphere MQmůžete použít tabulku CCDT (Client Channel Definition table) s třídami platformy .NET. Umístění tabulky CCDT určujete různými způsoby v závislosti na tom, zda používáte spravované nebo nespravované připojení.

Typ nespravovaného připojení klienta bez podpory XA nebo XA

V případě typu nespravovaného připojení můžete určit umístění tabulky CCDT dvěma způsoby:

- Pomocí proměnných prostředí MQCHLLIB určete adresář, kde se tabulka nachází, a MQCHLTAB pro určení názvu souboru tabulky.
- Použije se konfigurační soubor klienta. Ve stanze CHANNELS použijte atributy ChannelDefinitionAdresář k uvedení adresáře, kde se tabulka nachází, a ChannelDefinitionSoubor pro uvedení názvu souboru.

Je-li umístění určeno v konfiguračním souboru klienta i pomocí proměnných prostředí, proměnné prostředí budou mít prioritu. Tuto funkci můžete použít k určení standardního umístění v konfiguračním souboru klienta a k přepsání pomocí proměnných prostředí, je-li to nutné.

Typ připojení spravovaného klienta

Se spravovaným typem připojení můžete určit umístění tabulky CCDT třemi způsoby:

- Použití konfiguračního souboru aplikace .NET. V sekci CHANNELS použijte klíče ChannelDefinitionAdresář k určení adresáře, kde je tabulka umístěna, a ChannelDefinitionSoubor pro uvedení názvu souboru.
- Pomocí proměnných prostředí MQCHLLIB určete adresář, kde se tabulka nachází, a MQCHLTAB pro určení názvu souboru tabulky.
- Použije se konfigurační soubor klienta. Ve stanze CHANNELS použijte atributy ChannelDefinitionAdresář k uvedení adresáře, kde se tabulka nachází, a ChannelDefinitionSoubor pro uvedení názvu souboru.

Je-li umístění uvedeno více než jedním z těchto způsobů, proměnné prostředí mají přednost před konfiguračním souborem klienta a konfigurační soubor aplikace .NET má přednost před ostatními metodami. Tuto funkci můžete použít k určení standardního umístění v konfiguračním souboru klienta a přepsání nastavení pomocí proměnných prostředí nebo konfiguračního souboru aplikace, je-li to nutné.

Jak aplikace .NET určuje, jaká definice kanálu se má použít

V prostředí klienta WebSphere MQ .NET se definice kanálu, která má být použita, může 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, vybere se jedna z nich v následujícím pořadí priority:

1. Vlastnosti určené v konstruktoru MQQueueManager , ať už explicitně, nebo prostřednictvím zahrnutí *MQC.CHANNEL_PROPERTY* v tabulce hashtable vlastností
2. Vlastnost *MQC.CHANNEL_PROPERTY* v hašovací tabulce MQEnvironment.properties .

3. Vlastnost *Kanál* v MQEnvironment
4. Konfigurační soubor aplikace .NET, název sekce CHANNELS, klíč ServerConnectionParms (používá se pouze pro spravovaná připojení)
5. Proměnná prostředí *MQSERVER*
6. Konfigurační soubor klienta, stanza CHANNELS, Atribut ServerConnectionParms
7. Tabulka definic kanálů klienta (CCDT). Umístění tabulky CCDT je určeno v konfiguračním souboru aplikace .NET (platí pouze pro spravovaná připojení).
8. Tabulka definic kanálů klienta (CCDT). Umístění tabulky CCDT se zadává pomocí proměnných prostředí *MQCHLIB* a *MQCHLTAB* .
9. Tabulka definic kanálů klienta (CCDT). Umístění tabulky CCDT je určeno pomocí konfiguračního souboru klienta

U položek 1-3 je definice kanálu sestavena pole podle pole z hodnot poskytnutých aplikací. Tyto hodnoty mohou být poskytnuty pomocí různých rozhraní a pro každou z nich může existovat více hodnot. Hodnoty polí se přidávají do definice kanálu podle zadaného pořadí priority:

1. Hodnota položky *connName* v konstruktoru MQQueueManager
2. Hodnoty vlastností ze hašovací tabulky MQQueueManager.properties
3. Hodnoty vlastností hašovací tabulky MQEnvironment.properties
4. Hodnoty nastavené jako pole MQEnvironment (například MQEnvironment.Hostname, MQEnvironment.Port)

U položek 4-6 je jako hodnota dodána celá definice kanálu. Neurčená pole v definici kanálu mají výchozí nastavení systému. Žádné hodnoty z jiných metod definování kanálů a jejich polí se neslučují s těmito specifikacemi.

U položek 7-9 je celá definice kanálu převzata z tabulky CCDT. Pole, která nebyla výslovně uvedena, když byl kanál definován, mají výchozí nastavení systému. Žádné hodnoty z jiných metod definování kanálů a jejich polí se neslučují s těmito specifikacemi.

Použití kanálů kanálů v produktu IBM WebSphere MQ .NETTO

Pokud používáte vazby klienta, můžete použít uživatelské procedury kanálu jako pro jakékoli jiné připojení klienta. Používáte-li spravované vazby, musíte napsat uživatelský program, který implementuje příslušné rozhraní.

Vazby klienta

Pokud používáte vazby klienta, můžete použít uživatelské procedury kanálu, jak je popsáno v tématu Uživatelské procedury kanálu. Nelze použít uživatelské procedury kanálu zapsané pro spravované vazby.

Spravované vazby

Používáte-li spravované připojení k implementaci uživatelské procedury, definujete novou třídu .NET, která implementuje příslušné rozhraní. V balíku produktu WebSphere MQ jsou definována tříuživatelská rozhraní:

- MQSendExit
- MQReceiveExit
- MQSecurityExit

Poznámka: Uživatelské procedury zapsané pomocí těchto rozhraní nejsou v nespravovaném prostředí podporovány jako uživatelské procedury kanálu.

Následující ukázka definuje třídu, která implementuje všechny tři:

```
class MyMQExits : MQSendExit, MQReceiveExit, MQSecurityExit
{
```

```
   // This method comes from the send exit
  byte[] SendExit(MQChannelExit       channelExitParms,
                  MQChannelDefinition channelDefinition,
                  byte[]              dataBuffer
                  ref int             dataOffset
                  ref int             dataLength
                  ref int             dataMaxLength)
  {
     // complete the body of the send exit here
  }


  // This method comes from the receive exit
  byte[] ReceiveExit(MQChannelExit       channelExitParms,
                     MQChannelDefinition channelDefinition,
                     byte[]              dataBuffer
                     ref int             dataOffset
                     ref int             dataLength
                     ref int             dataMaxLength)
  {
     // complete the body of the receive exit here
  }


  // This method comes from the security exit
  byte[] SecurityExit(MQChannelExit       channelExitParms,
                      MQChannelDefinition channelDefParms,
                      byte[]              dataBuffer
                      ref int             dataOffset
                      ref int             dataLength
                      ref int             dataMaxLength)
  {
     // complete the body of the security exit here
  }

}
```

Každé uživatelské proceduře je předána funkce MQChannelExit a instance objektu MQChannelDefinition . Tyto objekty reprezentují struktury MQCXP a MQCD definované v procedurálním rozhraní.

Data, která mají být odeslána uživatelskou procedurou pro odeslání zprávy, a data přijatá v rámci zabezpečení nebo procedury příjmu, jsou určena pomocí parametrů uživatelské procedury.

U položky data na offsetu *dataOffset* s délkou *dataLength* v bajtovém poli *dataBuffer* jsou data, která mají být odeslána uživatelskou procedurou odeslání, a data přijatá v rámci zabezpečení nebo při ukončení příjmu. Parametr *dataMaxLength* poskytuje maximální délku (z *dataOffset*) dostupnou pro uživatelskou proceduru v *dataBuffer*. Poznámka: Pro uživatelskou proceduru zabezpečení je možné, aby dataBuffer měla hodnotu null, pokud se jedná o první zavolání ukončení procedury nebo ukončení partnera, který má odeslat žádná data.

Na oplátku by hodnota *dataOffset* a *dataLength* měla být nastavena tak, aby ukazovala na posun a délku v rámci vráceného pole bajtů, které by měly třídy platformy .NET používat. Pro uživatelskou proceduru odeslání to označuje data, která má odeslat, a pro uživatelskou proceduru pro zabezpečení nebo příjem dat, která by měla být interpretována. Ukončení by mělo normálně vrátit bajtové pole; výjimky jsou procedury zabezpečení, které se mohou rozhodnout neodesílat žádná data, a všechny uživatelské procedury volané s příčinami INIT nebo TERM. Nejjednodušší způsob výstupu, který lze zapsat, je tedy takový, který nedělá nic jiného než návrat dataBuffer:

Nejjednodušším možným výstupním tělem je:

```
{
  return dataBuffer;
}
```

Třída MQChannelDefinition

V 7.5.0.6 V prostředí produktu Version 7.5.0, Fix Pack 6je ID uživatele a heslo, které jsou zadány spolu se spravovanou aplikací klienta .NET, nastaveno ve třídě IBM WebSphere MQ .NET MQChannelDefinition , která je předána uživatelské proceduře zabezpečení klienta. Uživatelská procedura zabezpečení zkopíruje

ID uživatele a heslo na disk MQCD.RemoteUserIdentifier a MQCD.RemotePassword (viz “Psaní uživatelské procedury zabezpečení” na stránce 390).

Určení uživatelských procedur kanálu (spravovaného klienta)

Zadáte-li při vytváření objektu MQQueueManager název kanálu a název připojení (buď v konstruktoru MQEnvironment nebo v konstruktoru MQQueueManager), můžete kanály kanálu zadat dvěma způsoby.

V pořadí přednosti jsou tyto:

1. Předávání vlastností hašovací tabulky MQC.SECURITY_EXIT_PROPERTY, MQC.SEND_EXIT_PROPERTY nebo MQC.RECEIVE_EXIT_PROPERTY v konstruktoru MQQueueManager .
2. Nastavení vlastností MQEnvironment SecurityExit, SendExit nebo ReceiveExit .

Nezadáte-li název kanálu a název připojení, budou kanály kanálu používané k použití pocházející z definice kanálu převzaté z tabulky definic kanálů klienta (CCDT). Není možné přepsat hodnoty uložené v definici kanálu. Další informace o tabulkách definic kanálů naleznete v tématu Tabulka definic kanálů klienta a “Použití definiční tabulky kanálu klienta s rozhraním .NET” na stránce 566 .

V každém případě má specifikace tvar řetězce s následujícím formátem:

```
Assembly_name(Class_name)
```

název_třídy je úplný název, včetně specifikace oboru názvů, třídy platformy .NET, která implementuje IBM.WMQ.MQSecurityExit, IBM.WMQ.MQSendExit nebo IBM.WMQ.MQReceiveExit (podle potřeby). *název_jednotky sestavení* je úplné umístění sestavení, které obsahuje danou třídu, včetně přípony souboru. Délka řetězce je omezena na 999 znaků, používáte-li vlastnosti prostředí MQEnvironment nebo MQQueueManager. Je-li však název uživatelské procedury kanálu zadán v tabulce CCDT, je omezen na 128 znaků. Je-li to nezbytné, zavede kód klienta platformy .NET a vytvoří instanci uvedené třídy analýzou specifikace řetězce.

Určení uživatelských dat uživatelské procedury kanálu (spravovaného klienta)

Uživatelské procedury kanálu mohou mít k sobě přidružená uživatelská data. Pokud při vytváření objektu MQQueueManager zadáte název kanálu a název připojení (buď v konstruktoru MQEnvironment, nebo v konstruktoru MQQueueManager), můžete data uživatele zadávat dvěma způsoby.

V pořadí přednosti jsou tyto:

1. Předávání vlastností hašovací tabulky MQC.SECURITY_USERDATA_PROPERTY, MQC.SEND_USERDATA_PROPERTY nebo MQC.RECEIVE_USERDATA_PROPERTY v konstruktoru MQQueueManager .
2. Nastavení vlastností dat SecurityUserdat MQEnvironment, SendUserData nebo ReceiveUser.

Pokud nezadáte název kanálu a název připojení, použijí se hodnoty výstupních uživatelských dat, které mají být použity, pocházející z definice kanálu z tabulky CCDT (Client Channel Definition table). Není možné přepsat hodnoty uložené v definici kanálu. Další informace o tabulkách definic kanálů naleznete v tématu Tabulka definic kanálů klienta a “Použití definiční tabulky kanálu klienta s rozhraním .NET” na stránce 566 .

V každém případě je specifikace řetězec, omezen na 32 znaků.

Automatické opětovné připojení klienta v .NET

Můžete nastavit, aby se váš klient znovu automaticky připojil ke správci front během neočekávaného přerušení spojení.

Klient může být neočekávaně odpojen od správce front, pokud se například zastaví správce front nebo selže síť nebo server.

Bez automatického opětovného připojení klienta dojde k chybě při selhání připojení. Můžete použít kód chyby, který vám pomůže znovu ustanovit spojení.

Klient, který používá poskytovanou službu automatického připojení klienta, se nazývá znovu připojitelného klienta. Chcete-li vytvořit znovu připojitelného klienta, uveďte určité volby, které se nazývají volby opětovného připojení při připojování ke správci front.

Je-li klientská aplikace klientem produktu WebSphere MQ .NET, může se rozhodnout pro automatické opětovné připojení klienta zadáním příslušné hodnoty parametru CONNECT_OPTION_PROPERTY, když použijete třídu produktu `MQQueueManager` k vytvoření správce front. Podrobné informace o hodnotách CONNECT_OPTIONS_PROPERTY naleznete v tématu Volby opětovného připojení .

Můžete vybrat, zda se klientská aplikace vždy připojí a znovu připojí ke správci front stejného názvu, ke stejnému správci front nebo k jakékoli sadě správců front, které jsou definovány se stejným parametrem `QMNAME` v tabulce připojení klienta (podrobnosti viz Skupiny správců front v CCDT).

Podpora zabezpečení SSL (Secure Sockets Layer)

Následující oddíl se nevztahuje na spravovaného klienta.

Klientské aplikace WebSphere MQ pro klientské aplikace platformy .NET podporují šifrování SSL (Secure Sockets Layer). SSL poskytuje komunikační šifrování, ověření a integritu zpráv. Obvykle se používá k zabezpečení komunikace mezi dvěma rovnocennými partnery na Internetu nebo v rámci intranetu.

Povolení zabezpečení SSL

SSL je podporováno pouze pro připojení klienta. Chcete-li povolit zabezpečení SSL, musíte určit volbu CipherSpec , která má být použita při komunikaci se správcem front, a to musí odpovídat sadě CipherSpec nastaveným na cílovém kanálu.

Chcete-li povolit zabezpečení SSL, zadejte CipherSpec pomocí statické proměnné člena SSLCipherSpec MQEnvironment. Následující příklad se připojuje ke kanálu SVRCONN s názvem SECURE.SVRCONN.CHANNEL, která byla nastavena tak, aby vyžadovala zabezpečení SSL se sadou CipherSpec NULL_MD5:

```
MQEnvironment.Hostname         = "your_hostname";
MQEnvironment.Channel          = "SECURE.SVRCONN.CHANNEL";
MQEnvironment.SSLCipherSpec    = "NULL_MD5";
MQEnvironment.SSLKeyRepository = "C:\mqm\key";
MQQueueManager qmgr = new MQQueueManager("your_Q_manager");
```

Seznam CipherSpecsnajdete v části Určení CipherSpecs .

Vlastnost SSLCipherSpec může být také nastavena pomocí vlastnosti MQC.SSL_CIPHER_SPEC_PROPERTY v hašovací tabulce vlastností připojení.

Pro úspěšné připojení pomocí SSL musí být úložiště klíčů klienta nastaveno s kořenovým řetězcem certifikátů vydavatele certifikátů, ze kterého může být ověřován certifikát prezentovaný správcem front. Podobně, je-li vlastnost SSLClientAuth v kanálu SVRCONN nastavena na hodnotu MQSSL_CLIENT_AUTH_REQUIRED, musí úložiště klíčů klienta obsahovat identifikační osobní certifikát, kterému správce front důvěřuje.

Použití rozlišujícího názvu správce front

Správce front sám sebe identifikuje pomocí certifikátu SSL, který obsahuje *Rozlišovací jméno* (DN).

Aplikace klienta WebSphere MQ .NET může toto DN použít k ujištění, že komunikuje se správným správcem front. Vzorek DN se zadává pomocí proměnné názvu sslPeerproměnné MQEnvironment. Například nastavení:

```
MQEnvironment.SSLPeerName = "CN=QMGR.*, OU=IBM, OU=WEBSPHERE";
```

umožňuje úspěšné připojení k úspěchu pouze v případě, že správce front předloží certifikát s názvem Common Name začínajícím QMGR., a alespoň dva názvy organizačních jednotek, přičemž první z nich musí být IBM a druhý WEBSPHERE.

Vlastnost SSLPeerName může být také nastavena pomocí vlastnosti MQC.SSL_PEER_NAME_PROPERTY v hašovací tabulce vlastností připojení. Další informace o rozlišujících názvech a pravidlech pro nastavení názvů rovnocenných uzlů naleznete v části Zabezpečení.

Je-li parametr SSLPeerName nastaven, úspěšná připojení jsou úspěšná pouze v případě, že je nastavena na platný vzor a správce front představuje odpovídající certifikát.

Ošetření chyb při použití SSL

Následující kódy příčiny mohou být vydány třídami WebSphere MQ pro prostředí .NET při připojování ke správci front s použitím zabezpečení SSL:

MQRC_SSL_NOT_ALLOWED
Vlastnost SSLCipherSpec byla nastavena, ale bylo použito připojení vazeb. SSL podporuje pouze připojení klienta.

NESROVNALOST MQRC_SSL_PEER_NAME_
Vzorek DN určený ve vlastnosti SSLPeerName neodpovídal rozlišujícímu názvu představenému správcem front.

CHYBA MQRC_SSL_PEER_NAME_ERROR
Vzorek DN zadaný ve vlastnosti SSLPeerName není platný.

Použití programu .NET Monitor

Důležité informace naleznete v příručce Funkce, které lze použít pouze s primární instalací na systému Windows .

Monitor .NET Monitor je aplikace podobná monitoru spouštěčů produktu WebSphere MQ . Můžete vytvořit komponenty .NET, které jsou převedeny na instanci, kdykoli je přijata zpráva na monitorované frontě a která pak tuto zprávu zpracuje. Monitor .NET Monitor se spustí příkazem `runmqdnm` a zastaví se příkazem `endmqdnm` . Podrobné informace o těchto příkazech naleznete v příručce runmqdnm a endmqdnm.

Chcete-li použít .NET Monitor, napiš komponentu, která implementuje rozhraní IMQObjectTrigger , která je definovaná v souboru amqmdnm.dll.

Komponenty mohou být buď transakční, nebo netransakční. Transakční komponenta musí být zděděna z System.EnterpriseServices.ServicedComponent a musí být registrována buď jako RequiresTransaction , nebo SupportsTransaction. Nesmí být registrována jako RequiresNew , protože produkt .NET Monitor již zahájil transakci.

Komponenta přijímá objekty MQQueueManager, MQQueue a MQMessage z adresáře `runmqdnm`. Může také přijmout řetězec s parametry uživatele, pokud byl zadán, pomocí volby příkazového řádku *-u* , když bylo spuštěno `runmqdnm` . Všimněte si, že vaše komponenta přijímá obsah zprávy, která byla doručena do monitorované fronty v objektu MQMessage. Nemusí se připojovat ke správci front, otevřít frontu nebo získat zprávu samotnou. Komponenta musí poté zpracovat zprávu jako odpovídající a vrátit řízení do monitoru .NET.

Pokud byla vaše komponenta zapsána jako transakční komponenta, zaregistruje se k potvrzení nebo odvolání transakce pomocí funkcí poskytovaných produktem System.EnterpriseServices.ServicedComponent.

Protože komponenta přijímá objekty MQQueueManager a MQQueue, stejně jako zprávu, obsahuje informace o kontextu této zprávy a může například otevřít jinou frontu ve stejném správci front, aniž by bylo nutné samostatně připojit se k produktu WebSphere MQ.

Příklady fragmentů kódu

Toto téma obsahuje dva příklady komponent, které získají zprávu z produktu .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 WebSphere MQ .NET

Specické příkazy pro kompilaci aplikací .NET zapsaných v různých jazycích.

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

Chcete-li sestavit aplikaci C# pomocí tříd produktu WebSphere MQ pro prostředí .NET, použijte následující příkaz:

```
csc /t:exe /r:System.dll /r:amqmdnet.dll /lib:MQ_INSTALLATION_PATH\bin /out:MyProg.exe MyProg.cs
```

Chcete-li sestavit aplikaci ve Visual Basicu pomocí tříd produktu WebSphere MQ pro prostředí .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 tříd produktu WebSphere MQ pro prostředí .NET, použijte následující příkaz:

```
cl /clr MQ_INSTALLATION_PATH\bin Myprog.cpp
```

Pro ostatní jazyky si prohlédněte dokumentaci dodanou prodejcem jazyka.

Trasování programů WebSphere MQ .NET

V produktu WebSphere MQ .NET spustíte a můžete řídit prostředek trasování jako v programech WebSphere MQ pomocí modulu MQI.

Parametry -i a -p příkazu strmqtrc, které vám umožňují zadat identifikátory procesů a podprocesů a pojmenované procesy, však nemají žádný účinek.

Obvykle je třeba trasovací prostředek použít pouze na žádost služby IBM .

Informace o příkazech trasování najdete v tématu Použití trasování v systému Windows .

Vlastní kanál produktu IBM WebSphere MQ pro produkt Microsoft Windows Communication Foundation (WCF)

Vlastní kanál produktu Microsoft Windows Communication Foundation (WCF) pro produkt IBM WebSphere MQ odesílá a přijímá zprávy mezi klienty WCF a službami.

Související pojmy

"Úvod k použití vlastního kanálu produktu WebSphere MQ pro prostředí WCF s prostředím .NET 3" na stránce 575

Přehled informací dostupných pro programátory používající vlastní kanál produktu WebSphere MQ pro systém Windows Communication Foundation (WCF) s prostředím .NET 3.

"Použití vlastních kanálů produktu WebSphere MQ pro prostředí WCF" na stránce 579

Přehled informací dostupných pro programátory používající vlastní kanály produktu WebSphere MQ V7 pro systém Windows Communication Foundation (WCF).

"Použití ukázek WCF" na stránce 595

Ukázky produktu Okna Communication Foundation (WCF) poskytují některé jednoduché příklady použití vlastního kanálu produktu WebSphere MQ .

"Určování problémů s vlastním kanálem WCF pro produkt WebSphere MQ" na stránce 601

Pomocí trasování produktu WebSphere MQ můžete shromažďovat podrobné informace o tom, které různé části kódu produktu WebSphere MQ se budou provádět. Při použití služby WCF (Windows Communication Foundation) je generován samostatný výstup trasování pro trasování vlastního kanálu WCF, který je integrován s trasováním infrastruktury Microsoft WCF.

Úvod k použití vlastního kanálu produktu WebSphere MQ pro prostředí WCF s prostředím .NET 3

Přehled informací dostupných pro programátory používající vlastní kanál produktu WebSphere MQ pro systém Windows Communication Foundation (WCF) s prostředím .NET 3.

Jaký je vlastní kanál produktu WebSphere MQ pro prostředek WCF?

Vlastní kanál pro produkt WebSphere MQ je přenosový kanál s použitím unifikovaného programovacího modelu 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 používaných pro jejich připojení, což umožňuje použití alternativních přenosů nebo konfigurací v souladu s prostředím, ve kterém je implementována služba nebo aplikace.

Produkt WCF je spravován systémem Connections tak, že vytvoří zásobník kanálu obsahující požadovanou kombinaci:

- Protokolové prvky: Nepovinná sada prvků, kde žádný, jeden nebo více lze přidat do podpůrných protokolů, jako jsou standardy WS-*.
- Kodér zprávy: Povinný prvek v zásobníku, který řídí serializaci zprávy do svého formátu spoje.
- Transportní kanál: Povinný prvek v sadě, který je zodpovědný za přenos serializovaných zpráv do koncového bodu.

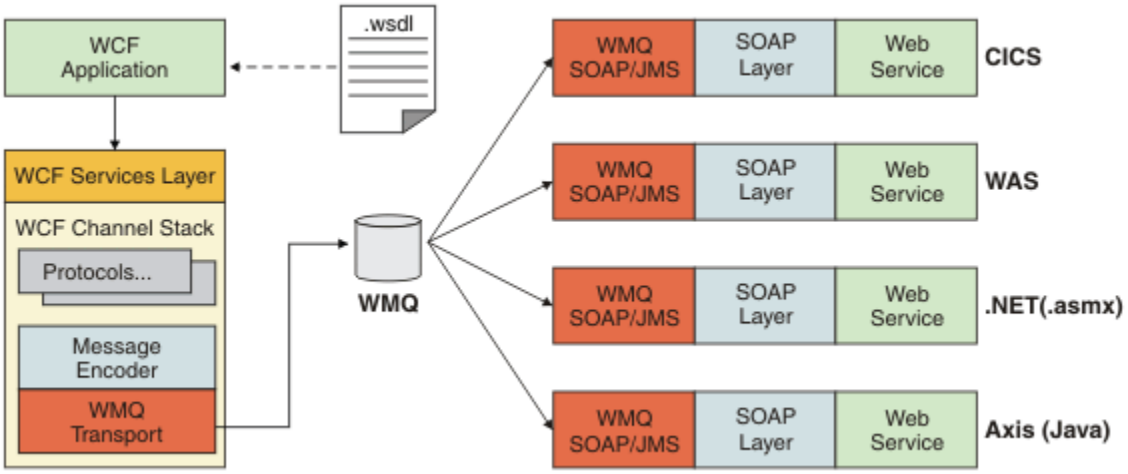
Vlastní kanál pro produkt WebSphere MQ je přenosový kanál a jako takový musí být spárován s kódovacím programem zpráv a s volitelnými protokoly, jak to požaduje aplikace používající vlastní vazbu WCF. Tímto způsobem mohou být aplikace, které byly vyvinuty pro použití služby WCF, pomocí vlastního kanálu pro produkt WebSphere MQ odesílat a přijímat data stejným způsobem, jako používají vestavěné přenosy poskytované společností Microsoft, a umožňují jednoduchou integraci s asynchronními, rozšiřitelnými a spolehlivými funkcemi systému zpráv produktu WebSphere MQ. Úplný seznam podporovaných funkcí viz: "Funkce a schopnosti vlastního kanálu služby WCF" na stránce 579.

Kdy a proč používám vlastní kanál produktu WebSphere MQ pro WCF?

Vlastní kanál produktu WebSphere MQ lze použít k odesílání a přijímání zpráv mezi klienty WCF a službami stejným způsobem, jako jsou vestavěné přenosy poskytované společností Microsoft, což umožňuje aplikacím přístup k funkcím produktu WebSphere MQ v rámci unifikovaného modelu programování WCF.

Typický scénář vzorku použití vlastního kanálu produktu WebSphere MQ pro WCF je jako rozhraní webových služeb hostovaných prostřednictvím produktu WebSphere MQ (SOAP/JMS)

Zprávy jsou přenášeny s použitím formátu zpráv protokolu SOAP prostřednictvím rozhraní JMS produktu WebSphere MQa umožňuje klientům a službám WCF volat nebo volat jiným aplikacím produktu WebSphere MQ nebo hostujícím prostředím kompatibilním s tímto formátem včetně webových služeb a klientů spuštěných v produktu WebSphere Application Server, CICS, Axis v1 (Java), a .asmx (.NET), jak je zobrazeno v následujícím diagramu:



Podrobné informace o protokolu SOAP prostřednictvím rozhraní JMS naleznete v tématu: “Přenos WebSphere MQ pro SOAP” na stránce 914

Příklad typického scénáře z diagramu by byl:

1. Webová služba provozovaná v rámci produktu WebSphere Application Server a vystavená prostřednictvím produktu WebSphere MQ s použitím podpory protokolu SOAP prostřednictvím rozhraní JMS na serveru WebSphere Application Server
2. Dokument WSDL popisující službu pak může nástroj WCF použít k vygenerování serveru proxy klienta a konfigurace, který by pak vytvořil příslušnou sadu kanálů WCF včetně vlastního kanálu.
3. Klientská aplikace pak může použít server proxy ke spuštění webové služby stejným způsobem jako jakákoli jiná webová služba.

Kanál by se obvykle použil s kódováním zpráv WCF text/SOAP, ale kanál může být v případě potřeby spárován s jinými enkodéry zpráv WCF. Použití alternativního kódovače může také poskytnout omezenou integraci s nativními aplikacemi produktu WebSphere MQ , které nepodporují protokol SOAP prostřednictvím rozhraní JMS, ale to není primární role kanálu.

Mezi klíčové výhody použití vlastního kanálu v prostředí WCF patří:

- Asynchronní vyvolání: Podpora operací fire fire a zapomenutí klienta, kde je klient oddělen od dostupnosti služby a funkcí, jako např. přesměrování odpovědí a vícesměrovacích uzlů.
- Spolehlivé charakteristiky škálování: Systém zpráv založený na frontě umožňuje předvídatelné přidání kapacity do systému.
- Kvalita služby: Zprávy jsou hmatatelné a vysledovatelné a lze je snadno spravovat a spravovat.

Softwarové požadavky a pokyny k instalaci vlastního kanálu produktu WebSphere MQ pro WCF

Toto téma popisuje požadavky na software a informace o instalaci pro vlastní kanál produktu WebSphere MQ pro WCF.

Vlastní kanál produktu WebSphere MQ pro prostředí WCF se může připojit pouze k produktu WebSphere MQ V7 nebo k vyšším správcům front.

Softwarové požadavky na vlastní kanál WCF pro produkt WebSphere MQ

V této části jsou uvedeny požadavky na software pro vlastní kanál WCF pro produkt WebSphere MQ.

Běhové prostředí

- Na hostitelském počítači musí být nainstalován produkt Microsoft .NET Framework v3.0 nebo vyšší.
- *Java a .NET Messaging and Web Services* se standardně instaluje jako součást instalačního programu produktu WebSphere MQ 7.0.1 . Nainstaluje pro vlastní kanál sestavení prostředí .NET potřebné pro vlastní kanál do mezipaměti Global Assembly Cache.

Poznámka: Pokud před instalací produktu WebSphere MQ V7.0.1není nainstalován produkt Microsoft .NET Framework v2.0 nebo vyšší, instalace produktu WebSphere MQ bude pokračovat bez chyby, ale vlastní kanál produktu WebSphere MQ je nedostupný. Je-li prostředí .NET Framework nainstalováno po instalaci produktu WebSphere MQ 7.0.1, je nutné aktivovat vlastní kanál produktu WebSphere MQ spuštěním skriptu *WMQInstallDir*\bin\amqiRegisterdotNet.cmd , kde *WMQInstallDir* je adresář, ve kterém je nainstalován produkt WebSphere MQ 7.0.1 . Tento skript nainstaluje vyžadované sestavení v mezipaměti GAC (Global Assembly Cache). Sada souborů amqi*.log zaznamenávající provedené akce se vytvoří v adresáři %TEMP% . It is not necessary to rerun the amqiRegisterdotNet.cmd script if .NET is upgraded to v3.0 or higher from an earlier version, for example, from .NET v2.0.

Vývojové prostředí

- Microsoft Visual Studio 2008 nebo Windows Software Development Kit for .NET 3.0 nebo novější.
- Na hostitelském počítači musí být nainstalován produkt Microsoft .NET Framework V3.5 nebo vyšší, aby bylo možné sestavit ukázkové soubory řešení.

Poznámka: Pokud před instalací produktu WebSphere MQ V7.0.1není nainstalován produkt Microsoft .NET Framework v2.0 nebo vyšší, instalace produktu WebSphere MQ bude pokračovat bez chyby, ale vlastní kanál produktu WebSphere MQ je nedostupný. Je-li prostředí .NET Framework nainstalováno po instalaci produktu WebSphere MQ 7.0.1, je nutné aktivovat vlastní kanál produktu WebSphere MQ spuštěním skriptu *WMQInstallDir*\bin\amqiRegisterdotNet.cmd , kde *WMQInstallDir* je adresář, ve kterém je nainstalován produkt WebSphere MQ 7.0.1 . Tento skript nainstaluje vyžadované sestavení v mezipaměti GAC (Global Assembly Cache). Sada souborů amqi*.log zaznamenávající provedené akce se vytvoří v adresáři %TEMP% . It is not necessary to rerun the amqiRegisterdotNet.cmd script if .NET is upgraded to v3.0 or higher from an earlier version, for example, from .NET v2.0.

Vlastní kanál produktu WebSphere MQ pro prostředek WCF: Co je nainstalováno?

Vlastní kanál pro produkt WebSphere MQ je přenosový kanál s použitím unifikovaného programovacího modelu Microsoft Windows Communication Foundation (WCF). Vlastní kanál je standardně instalován jako součást instalace produktu WebSphere MQ 7.0.1 .

Vlastní kanál produktu WebSphere MQ pro prostředek WCF

Vlastní kanál produktu WebSphere MQ pro komponentu WCF je standardně instalován jako součást instalace produktu WebSphere MQ 7.0.1 ; vlastní kanál a jeho závislosti jsou obsaženy v komponentě

produktu Java and .NET Messaging and Web Services , která je standardně instalována. Při upgradu na produkt WebSphere MQ 7.0.1 ze starší verze bude aktualizace při výchozím nastavení instalovat vlastní kanál produktu WebSphere MQ pro produkt WCF, pokud již byla dříve nainstalována komponenta Java and .NET Messaging and Web Services v dřívější instalaci.

Komponenta Java and .NET Messaging and Web Services obsahuje soubor IBM.XMS.WCF.dll a soubor IBM.XMS.WCF.dll je hlavním vlastním sestavením kanálu, který obsahuje třídy rozhraní WCF. Tento soubor je nainstalován v mezipaměti GAC (Global Assembly Cache) a je k dispozici také v následujícím adresáři: *MQ_INSTALLATION_PATH*\bin , kde *MQ_INSTALLATION_PATH* je adresář, ve kterém je nainstalován produkt WebSphere MQ 7.0.1 .

Klíčové třídy vyžadované pro použití vlastního kanálu jsou uvedeny v poli *Obor názvů: IBM.XMS.WCF* a:

Název vazby přenosu	IBM.XMS.WCF.SoapJmsIbmTransportBindingElement
Vazba importu vazeb	IBM.XMS.WCF.SoapJmsIbmTransportBindingElementImporter
Konfigurace vazby přenosu	IBM.XMS.WCF.SoapJmsIbmTransportBindingElementConfig

Ukázky vlastního kanálu produktu WebSphere MQ

Ukázky poskytují některé jednoduché příklady použití vlastního kanálu produktu WebSphere MQ pro prostředí WCF. Ukázky a jejich přidružené soubory jsou umístěny v adresáři *MQ_INSTALLATION_PATH*\tools\wcf\samples\ , kde *MQ_INSTALLATION_PATH* je instalační adresář produktu WebSphere MQ. Další informace o ukázkách vlastních kanálů produktu WebSphere MQ naleznete v následujícím tématu: “Použití ukázek WCF” na stránce 595

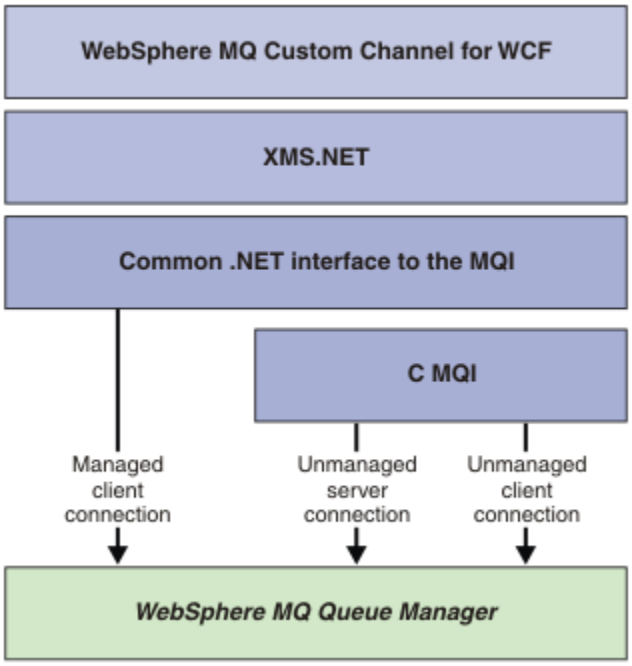
svcutil.exe.config

svcutil.exe.config je příkladem nastavení konfigurace vyžadovaného k povolení nástroje pro generování klienta proxy klienta Microsoft WCF svcutil k rozpoznání vlastního kanálu. Soubor svcutil.exe.config je umístěn v adresáři *MQ_INSTALLATION_PATH*\tools\wcf\docs\examples\ , kde *MQ_INSTALLATION_PATH* je instalační adresář produktu WebSphere MQ. Další informace o použití příkazu svcutil.exe.confignaleznete v následujícím tématu: “Generování serveru proxy klienta WCF a konfiguračních souborů aplikace pomocí nástroje svcutil s metadaty ze spuštěné služby” na stránce 592.

Architektura WCF

Vlastní kanál produktu WebSphere MQ pro WCF je integrován v rozhraní API produktu IBM Message Service Client for .NET (XMS .NET).

Architektura WCF je zobrazena v následujícím diagramu:



Všechny vyžadované komponenty jsou standardně instalovány spolu s instalací produktu WebSphere MQ V7.0.1 .

Tři připojení jsou: Připojení spravovaného klienta, Nespravovaná připojení k serveru a Nespravovaná připojení klienta. Další informace o těchto spojeních najdete v tématu “Volby připojení WCF” na stránce 583.

Použití vlastních kanálů produktu WebSphere MQ pro prostředí WCF

Přehled informací dostupných pro programátory používající vlastní kanály produktu WebSphere MQ V7 pro systém Windows Communication Foundation (WCF).

Produkt Microsoft Windows Communication Foundation je založen na webových službách a podpoře systému zpráv v prostředí Microsoft .NET Framework 3. Produkt WebSphere MQ V7 lze nyní používat jako vlastní kanál v rámci WCF v rámci platformy .NET Framework 3 stejným způsobem jako vestavěné kanály nabízené společností Microsoft.

Zprávy, které jsou přepravovány přes vlastní kanál, jsou formátovány podle implementace protokolu SOAP prostřednictvím rozhraní JMS produktu WebSphere MQ V7. Aplikace pak mohou komunikovat se službami hostovanými WCF nebo s infrastrukturou služeb WebSphere SOAP over JMS. Podrobné informace o protokolu SOAP prostřednictvím rozhraní JMS naleznete v tématu: “Přenos WebSphere MQ pro SOAP” na stránce 914

Funkce a schopnosti vlastního kanálu služby WCF

V následujících tématech naleznete informace o funkcích a schopnostech vlastního kanálu služby WCF.

Vlastní tvary kanálů WCF

Přehled vlastních tvarů kanálů, které lze použít v produktu WebSphere MQ jako v rámci vlastních kanálů Microsoft Windows Communication Foundation (WCF).

Vlastní kanál produktu WebSphere MQ pro prostředí WCF podporuje dva tvary kanálů:

- Jednosměrná
- Požadavek-odpověď

WCF automaticky vybere tvar kanálu podle hostované smlouvy služby.

Zakázky, které zahrnují metody, které používají pouze parametr **IsOneWay** , jsou obsluhovány jednosměrným tvarem kanálu, například:

```
[OperationContract(IsOneWay = true)]
void printString(String text);
```

Smlouvy, které zahrnují buď směs jednocestné a typu požadavek-odezva, nebo všechny metody požadavek-odpověď, jsou obsluhovány ve tvaru kanálu požadavek-odezva. Příklad:

```
[OperationContract]
int subtract(int a, int b);

[OperationContract(IsOneWay = true)]
void printString(string text);
```

Poznámka: Míšujete-li jednosměrné metody a metody požadavek-odezva ve stejné smlouvě, musíte zajistit, aby chování bylo zamýšleno, zejména při práci ve smíšeném prostředí, protože jednosměrné metody čekají, dokud neobdrží od služby odpověď s hodnotou null.

Jednosměrný kanál

Vlastní kanál WebSphere MQ pro WCF se používá, například k odesílání zpráv z klienta WCF pomocí tvaru jednosměrného kanálu. Kanál může odesílat zprávy pouze jedním směrem, například ze správce front klienta do fronty ve službě WCF.

Kanál požadavků požadavku

Vlastní kanál požadavku WebSphere MQ -odpověď pro WCF se používá například k asynchronnímu odesílání zpráv ve dvou směrech; pro asynchronní zasílání zpráv musí být použita stejná instance klienta. Kanál může odesílat zprávy jedním směrem, například od správce front klienta do fronty v rámci služby WCF a poté odeslat zprávu odpovědi z WCF do fronty ve správci front klienta.

Názvy a hodnoty parametrů identifikátoru URI WCF

connectionFactory
Parametr connectionFactory je povinný. Syntaxe tohoto parametru naleznete v tématu Syntaxe identifikátoru URI a parametry implementace webové služby .

InitialContextFactory
Parametr továrny initialContextje povinný a musí být nastaven na hodnotu "com.ibm.mq.jms.Nojndi" kvůli kompatibilitě se serverem WebSphere Application Server a dalšími produkty (viz “Implementace služby na server WebSphere Application Server pro použití přenosu WebSphere pro protokol SOAP” na stránce 972).

Doručení vlastního kanálu WCF

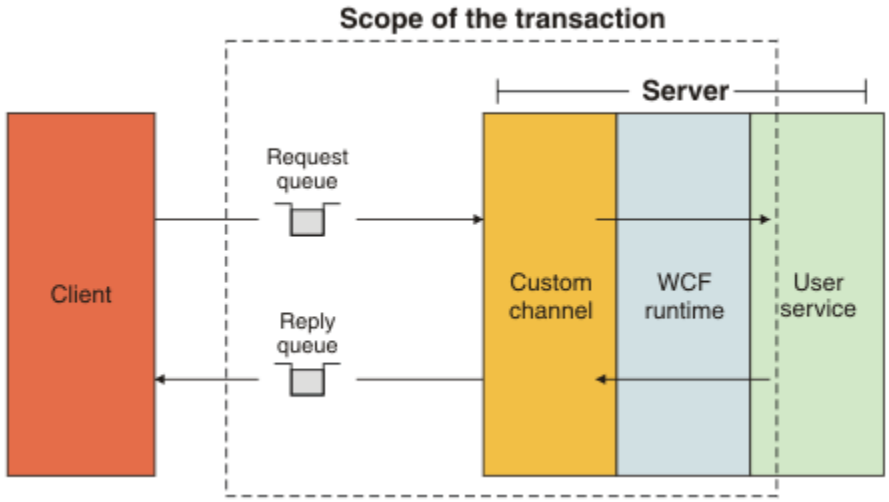
Zajištěné doručení zaručuje, že požadavek na službu nebo odpověď se bude provádět a nebude ztracena.

Je přijata zpráva požadavku a každá zpráva odpovědi se odešle pod bodem synchronizace lokální transakce, která může být odvolána v případě selhání běhového prostředí. Příklady těchto selhání jsou: Neošetřená výjimka vyvolaná službou, selhání při odeslání zprávy do služby nebo selhání doručení zprávy odpovědi.

`AssuredDelivery` je atribut zajištěného doručení, který lze uvést na servisní smlouvě, aby zaručil, že všechny zprávy vzniklé při zpracování požadavku přijaté službou a každá zpráva odpovědi odeslané ze služby se neztratí v případě selhání za běhu.

Aby se zajistilo, že zprávy budou zachovány i v případě selhání systému nebo výpadku proudu, musí být zprávy odeslány jako trvalé. Chcete-li používat trvalé zprávy, musí mít klientská aplikace tuto volbu určenou na svém identifikátoru URI koncového bodu. Další informace o nastavení vlastností identifikátoru URI viz: Syntaxe identifikátoru URI a parametry pro implementaci webové služby.

Distribuované transakce nejsou podporovány a rozsah transakce se nerozšiřuje nad zpracování zpráv požadavků a odpovědí provedených produktem WebSphere MQ. Jakákoli práce provedená v rámci služby může být znovu spuštěna jako výsledek selhání, který způsobí, že zpráva bude přijata znovu. Následující diagram zobrazuje rozsah transakce:



Zajištěné doručení je povoleno použitím atributu `AssuredDelivery` pro třídu služeb, jak je zobrazeno v následujícím příkladu:

```
[AssuredDelivery]
class TestCalculatorService : IWMQSampleCalculatorContract
{
    public int add(int a, int b)
    {
        int ans = a + b;
        return ans;
    }
}
```

Používáte-li atribut `AssuredDelivery` , musíte mít na paměti následující body:

- Když kanál zjistí, že se selhání bude opakovat, pokud byla zpráva odvolána a přijata znovu, zpráva se bude považovat za nezpracovatelnou zprávu a nevrátí se do fronty požadavků na přepracování. Například: Pokud přijatá zpráva není správně formátovaná nebo nemůže být odeslána do služby. Neošetřené výjimky vyvolané z operace služby jsou vždy znovu odeslány, dokud nebyla zpráva znovu doručena do maximálního počtu, který je uvedený ve vlastnosti prahové hodnoty vrácení fronty požadavků. Další informace viz: “nezpracovatelné zprávy kanálu vlastního kanálu WCF” na stránce 582
- Kanál provádí čtení, zpracování a odpovědi každé zprávy s požadavkem jako atomickou operaci pomocí jednoho podprocesu provádění k vynucení integrity transakcí. Chcete-li povolit spouštění operací služby souběžně, kanál umožňuje serveru WCF vytvořit více instancí kanálu. Počet instancí kanálu, které jsou k dispozici pro zpracování požadavků, je řízen vlastností vazby `MaxConcurrentCalls`. Další informace viz: “Volby konfigurace vazby WCF” na stránce 588
- Funkce assured delivery používá jak IOperationInvoker , tak i IErrorHandler -body rozšiřitelnosti WCF. Pokud jsou tyto body rozšiřitelnosti používány externě aplikací, aplikace musí zajistit, aby byly volány všechny dříve registrované rozšiřitelné body. Pokud tak neučiníte pro IErrorHandler , může dojít k neohlášenému hlášení chyb. Pokud tak neučiníte, může produkt IOperationInvoker způsobit, že se WCF přestane reagovat.

Vlastní zabezpečení kanálu WCF

Vlastní kanál produktu WebSphere MQ pro službu WCF podporuje použití zabezpečení SSL pouze pro nespravovaná připojení klienta ke správci front.

SSL může být uvedeno jedním ze dvou způsobů:

- Určete SSL přímo na identifikátoru URI protokolu SOAP prostřednictvím rozhraní JMS. Úplný popis voleb SSL viz SSL a přenos WebSphere MQ pro SOAP .

- Uveďte SSL pomocí položky v tabulce definic kanálů klienta (CCDT). Další informace o CCDT viz Tabulka definic kanálů klienta

Tabulky definic kanálů klienta WCF (CCDT)

Vlastní kanál produktu WebSphere MQ pro prostředí WCF podporuje použití tabulek CCDT (Client Channel Definition CCDT) ke konfiguraci informací o připojení pro klientská připojení.

CCDTs jsou řízeny těmito dvěma proměnnými prostředí:

- *MQCHLLIB* určuje adresář, ve kterém je umístěna tabulka.
- *MQCHLTAB* určuje název souboru tabulky.

Tabulku definic kanálů nelze určit přímo v identifikátoru URI protokolu SOAP prostřednictvím rozhraní JMS. Jsou-li tyto proměnné prostředí definovány, pak mají přednost před všemi podrobnostmi o připojení klienta uvedenými v identifikátoru URI.

Další informace o tabulkách definic kanálů klienta naleznete v tématu: Tabulka definic kanálů klienta .

Související pojmy

Tabulka definic kanálů klienta

nezpracovatelné zprávy kanálu vlastního kanálu WCF

Pokud služba selže při zpracování zprávy požadavku nebo selže doručení zprávy odpovědi do fronty odpovědí, bude zpráva považována za nezpracovatelnou zprávu.

Nezpracovatelné zprávy požadavku

Pokud zprávu požadavku nelze zpracovat, bude s ní zacházeno jako s nezpracovatelnou zprávou. Tato akce zabrání tomu, aby služba znovu obdržela stejnou znovu zpracovatelnou zprávu. Pro nezpracovatelnou zprávu požadavku, která má být považována za nezpracovatelnou zprávu, musí být splněna jedna z následujících situací:

- Počet vrácení zpráv překročil prahovou hodnotu odvolání uvedenou ve frontě požadavků, která se vyskytne pouze, pokud byla pro službu uvedena zajištěná doručení. Další informace o zajištěného doručení najdete v tématu: “Doručení vlastního kanálu WCF” na stránce 580
- Zpráva nebyla správně naformátována a nelze ji interpretovat jako zprávu protokolu SOAP prostřednictvím rozhraní JMS.

Nezpracovatelné zprávy odpovědi

Pokud služba nedoručí zprávu odpovědi do fronty odpovědí, je zpráva odpovědi považována za nezpracovatelnou zprávu. U zpráv s odpovědí umožňuje tato akce později získat zprávy odpovědí za účelem určení příčiny pomoci.

Zpracování nezpracovatelných zpráv

Akce provedená pro nezpracovatelnou zprávu závisí na konfiguraci správce front a na hodnotách nastavených ve volbách sestavy ve zprávě. Pro protokol SOAP prostřednictvím rozhraní JMS jsou ve výchozím nastavení nastaveny následující volby sestavy na zprávách požadavků a nejsou konfigurovatelné:

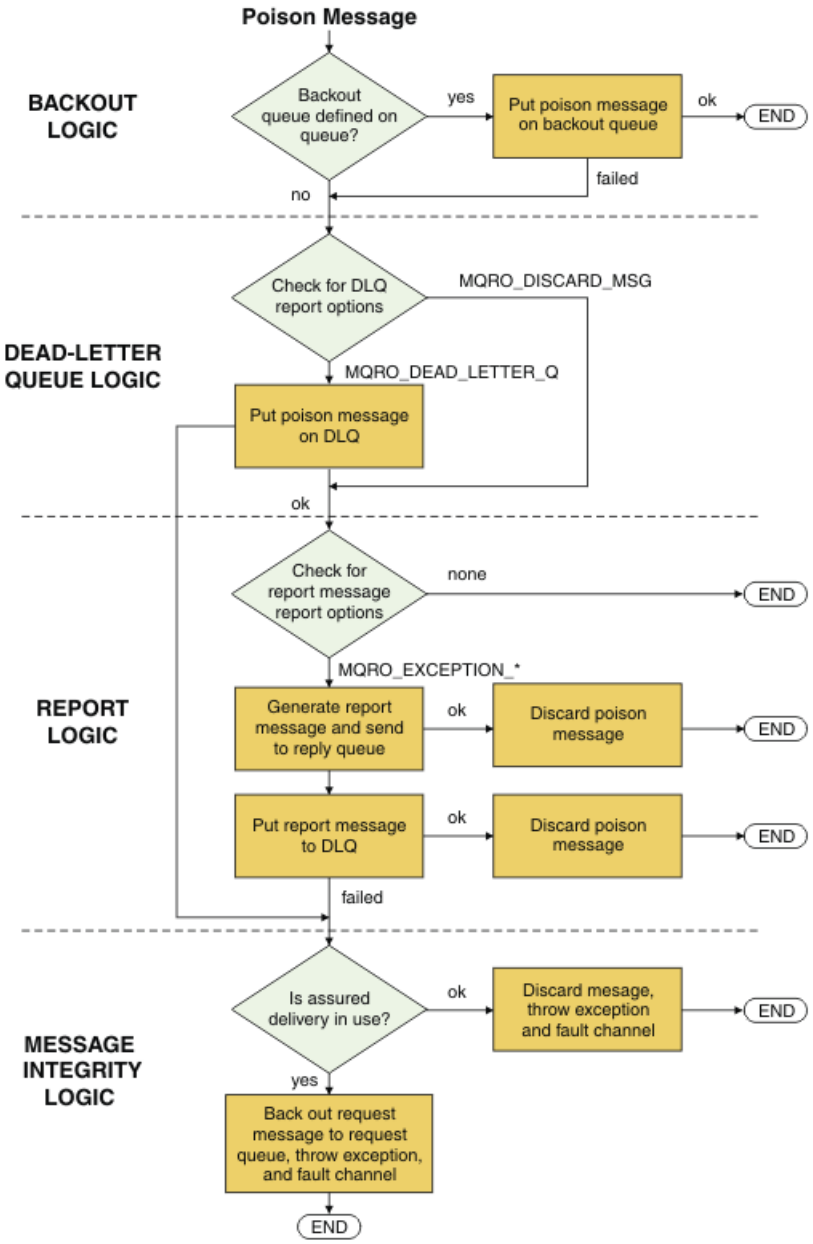
- MQRO_EXCEPTION_WITH_FULL_DATA
- MQRO_EXPIRATION_WITH_FULL_DATA
- MQRO_DISCARD_MSG

Pro protokol SOAP prostřednictvím rozhraní JMS je ve výchozím nastavení pro zprávy odpovědí nastavena následující volba sestavy a není konfigurovatelná:

- MQRO_DEAD_LETTER_Q

Pokud zprávy pocházejí ze zdroje mimo WCF, odkazujte se na dokumentaci pro tento zdroj.

Následující diagram zobrazuje možné akce a kroky, které podnikli v případě selhání zpracování nezpracovatelných zpráv:



Volby připojení WCF

K dispozici jsou tři režimy připojení vlastního kanálu produktu WebSphere MQ pro správce front WCF ke správci front. Zvažte, který typ připojení nejlépe vyhovuje vašim požadavkům.

Další informace o volbách připojení naleznete v tématu: “Rozdíly spojení” na stránce 555

Další informace o architektuře WCF naleznete v tématu: “Architektura WCF” na stránce 578

Nespravované připojení klienta

Připojení vytvořené v tomto režimu se připojuje jako klient WebSphere MQ k serveru WebSphere MQ spuštěnému buď na lokálním počítači, nebo na vzdáleném počítači.

Chcete-li použít vlastní kanál WebSphere MQ pro WCF jako klienta WebSphere MQ , můžete jej nainstalovat pomocí klienta WebSphere MQ MQI, a to buď na serveru WebSphere MQ , nebo na samostatném počítači.

Nespravované připojení k serveru

Při použití v režimu vázání serveru používá vlastní kanál produktu WebSphere MQ pro službu WCF rozhraní API správce front, a nikoli komunikace prostřednictvím sítě. Použití připojení vazeb poskytuje lepší výkon pro aplikace WebSphere MQ než použití síťových připojení.

Chcete-li používat připojení vazeb, je třeba instalovat vlastní kanál produktu WebSphere MQ pro prostředek WCF na serveru WebSphere MQ .

Připojení spravovaného klienta

Připojení vytvořené v tomto režimu se připojuje jako klient WebSphere MQ k serveru WebSphere MQ spuštěnému buď na lokálním počítači, nebo na vzdáleném počítači.

Vlastní třídy kanálu produktu WebSphere MQ pro připojení .NET 3 v tomto režimu zůstávají ve spravovaném kódu .NET a nevolají po nativních službách. Další informace o spravovaném kódu naleznete v dokumentaci společnosti Microsoft .

Pro použití spravovaného klienta existuje několik omezení. Další informace o těchto omezeních viz "Připojení spravovaného klienta" na stránce 556.

Vytvoření a konfigurace vlastního kanálu produktu WebSphere MQ pro WCF

Vlastní kanály produktu WebSphere MQ V7 pro funkci WCF pracují stejným způsobem jako kanály služby WCF nabízené společností Microsoft. Vlastní kanál produktu WebSphere MQ pro prostředí WCF lze vytvořit jedním ze dvou způsobů.

Informace o této úloze

Vlastní kanál produktu WebSphere MQ se integruje s kanálem WCF jako přenosový kanál WCF a jako takový musí být spárován s kódováním zpráv a volitelnými kanály protokolu, takže může vytvořit úplnou sadu kanálů, kterou může aplikace použít. Pro úspěšné vytvoření úplné sady kanálů jsou vyžadovány dva prvky:

1. Definice vázání: Určuje, které prvky jsou vyžadovány pro sestavení sady kanálů aplikací, včetně transportního kanálu, kódování zpráv a jakýchkoli protokolů a veškerých obecných nastavení konfigurace. Pro vlastní kanál musí být definice vazby vytvořena ve formě přizpůsobené vazby WCF.
2. Definice koncového bodu: Odkazuje na smlouvu na službu s definicí vazby a také poskytuje skutečný identifikátor URI připojení, který popisuje, kde se aplikace může připojit. Pro vlastní kanál je identifikátor URI ve formě identifikátoru URI protokolu SOAP prostřednictvím rozhraní JMS.

Tyto definice lze vytvořit jedním ze dvou způsobů:

- Administrativně; Definice se vytvářejí poskytnutím podrobností v konfiguračním souboru aplikace (například: `app.config`).
- Programové; Definice jsou vytvářeny přímo z kódu aplikace.

Rozhodnutí o tom, jaká metoda použít k vytvoření definic musí být založena na požadavcích na aplikaci:

- Administrativní metoda konfigurace poskytuje flexibilitu pro změnu podrobností o službách a klientu po implementaci bez opětovného sestavení aplikace.
- Programová metoda pro konfiguraci poskytuje větší ochranu před chybami konfigurace a schopnost dynamicky generovat konfiguraci za běhu.

Vytvoření vlastního uživatelského kanálu WCF administrativně dodáním informací o vazbách a koncových bodech v konfiguračním souboru aplikace

Vlastní kanál produktu WebSphere MQ pro prostředek WCF je kanálem WCF na úrovni transportu. Aby bylo možné použít vlastní kanál, musí být definován koncový bod a vazba a tyto definice lze provést dodáním informací o vazbách a koncových bodech v konfiguračním souboru aplikace.

Chcete-li konfigurovat a používat vlastní kanál produktu WebSphere MQ pro službu WCF, což je kanál WCF na úrovni transportu, musí být definována vazba a definice koncového bodu. Vazba obsahuje informace

o konfiguraci kanálu a definice koncového bodu obsahuje podrobnosti o připojení. Tyto definice lze vytvořit dvěma způsoby:

- Programově přímo z kódu aplikace, jak je popsáno zde: "Vytvoření vlastního kanálu WCF nahrazením informací o vazbách a koncovém bodu programově" na stránce 586
- Administrativně tak, že poskytnete podrobnosti v konfiguračním souboru aplikace, jak je popsáno v následujícím postupu.

Konfigurační soubor aplikace klienta nebo služby se běžně nazývá *yourappname*`.exe.config` , kde *yourappname* je název vaší aplikace. Konfigurační soubor aplikace je nejsnáze upravován s použitím nástroje Microsoft Configuration Editor s názvem `SvcConfigEditor.exe` následujícím způsobem:

- Spusťte nástroj editoru konfigurace produktu `SvcConfigEditor.exe` . Výchozí umístění instalace pro nástroj je: *`Drive:`*`\Program Files\Microsoft SDKs\Windows\v6.0\Bin\SvcConfigEditor.exe` , kde *Drive:* je název instalační jednotky.

Krok 1: Přidání rozšíření prvku vazby k povolení prostředku WCF k vyhledání vlastního kanálu

1. Klepnutím na volbu **Rozšířené** > **Rozšíření** > **prvek vazby** otevřete nabídku a vyberte volbu **Nový** .
2. Vyplňte pole, jak je uvedeno v této tabulce:

Tabulka 73. Nová pole prvku vazby

Pole	**Hodnota**
Název	`IBM.XMS.WCF.SoapJmsIbmTransportChannel`
Typ	Přejděte do produktu `IBM.XMS.WCF.dll` v mezipaměti GAC (Global Assembly Cache) a vyberte volbu `IBM.XMS.WCFSoapJmsIbmTransportBindingElementConfig` .

Krok 2: Vytvoření vlastní definice vázání, která páruje vlastní kanál s kódováním zpráv WCF

1. Klepnutím pravým tlačítkem myši na volbu **Vazby** otevřete nabídku a vyberte volbu **Nová konfigurace vazby** .
2. Vyplňte pole, jak je uvedeno v této tabulce:

Tabulka 74. Nová pole konfigurace vazby

Pole	**Hodnota**
Název	`CustomBinding_WMQ`
BindingElement 1	`textMessageEncoding (MessageVersion: Soap11)`
BindingElement 2	`IBM.XMS.WCF.SoapJmsIbmTransportChannel`

Krok 3: Určete vlastnosti vázání

1. Vyberte *IBM.XMS.WCF.SoapJmsIbmTransportChannel* vazba přenosu z vazby, kterou jste vytvořili v: "Krok 2: Vytvoření vlastní definice vázání, která páruje vlastní kanál s kódováním zpráv WCF" na stránce 585
2. Proveďte všechny požadované změny výchozích hodnot vlastností, jak je popsáno v: "Volby konfigurace vazby WCF" na stránce 588

Krok 4: Vytvoření definice koncového bodu

Vytvořte definici koncového bodu, která odkazuje na vlastní vazbu, kterou jste vytvořili v: "Krok 2: Vytvoření vlastní definice vázání, která páruje vlastní kanál s kódováním zpráv WCF" na stránce 585 , a poskytuje podrobnosti o připojení služby. Způsob, jakým jsou tyto informace zadány, závisí na tom, zda je definice pro klientskou aplikaci nebo pro aplikaci služeb.

Pro klientskou aplikaci přidejte definici koncového bodu do sekce klienta následujícím způsobem:

1. Klepněte pravým tlačítkem myši na nabídku **Klient** > **Koncové body** , abyste otevřeli nabídku a vyberte volbu **Nový koncový bod klienta** .
2. Vyplňte pole, jak je uvedeno v této tabulce:

Tabulka 75. Pole nového koncového bodu klienta

Pole	**Hodnota**
Název	`Endpoint_WMQ`
Adresa	*Identifikátor URI SOAP/JMS popisující podrobnosti připojení WMQ vyžadované pro přístup ke službě.* Další podrobnosti viz: "Vlastní kanál produktu WebSphere MQ pro formát adresy identifikátoru URI koncového bodu služby WCF" na stránce 587
Vazba	`customBinding`
BindingConfiguration	`CustomBinding_WMQ`
Smlouva.	*Název rozhraní smlouvy služeb*

Pro aplikaci služby přidejte do sekce služeb definici služby takto:

1. Klepnutím na tlačítko **Služby** otevřete nabídku a vyberte volbu **Nová služba**a poté vyberte třídu služeb, kterou chcete hostovat.
2. Přidejte definici koncového bodu do sekce **Koncové body** pro novou službu a doplňte pole tak, jak je uvedeno v této tabulce:

Tabulka 76. Nová pole koncového bodu služby

Pole	**Hodnota**
Název	`Endpoint_WMQ`
Adresa	*Identifikátor URI SOAP/JMS popisující podrobnosti připojení WMQ vyžadované pro přístup ke službě.* Další podrobnosti viz: "Vlastní kanál produktu WebSphere MQ pro formát adresy identifikátoru URI koncového bodu služby WCF" na stránce 587
Vazba	`customBinding`
BindingConfiguration	`CustomBinding_WMQ`
Smlouva.	*Název implementační třídy služby*

Vytvoření vlastního kanálu WCF nahrazením informací o vazbách a koncovém bodu programově

Vlastní kanál produktu WebSphere MQ pro prostředek WCF je kanálem WCF na úrovni transportu. Koncový bod a vazba musí být definovány pro použití vlastního kanálu a tyto definice lze provádět programově přímo z kódu aplikace.

Chcete-li konfigurovat a používat vlastní kanál produktu WebSphere MQ pro službu WCF, což je kanál WCF na úrovni transportu, musí být definována vazba a definice koncového bodu. Vazba obsahuje informace o konfiguraci kanálu a definice koncového bodu obsahuje podrobnosti o připojení. Další informace viz: “Použití ukázek WCF” na stránce 595

Tyto definice lze vytvořit dvěma způsoby:

- Administrativně tím, že poskytnete podrobnosti v konfiguračním souboru aplikace, jak je popsáno zde: “Vytvoření vlastního uživatelského kanálu WCF administrativně dodáním informací o vazbách a koncových bodech v konfiguračním souboru aplikace” na stránce 584
- Programově přímo z kódu aplikace, jak je popsáno v následujícím příkladu.

Krok 1: Vytvoření instance prvku vazby přenosu kanálu

Přidejte do své aplikace tento kód:

```
SoapJmsIbmTransportBindingElement transportBindingElement = new
SoapJmsIbmTransportBindingElement();
```

Krok 2: Nastavení vlastností vazby

Nastavte všechny vyžadované vlastnosti vazby, například přidáním následujícího kódu do vaší aplikace pro nastavení produktu `ClientConnectionMode`.

```
transportBindingElement.ClientConnectionMode = XmsWCFBindingProperty.AS_URI;
```

Krok 3: Vytvořte vlastní vazbu, která bude obsahovat dvojici transportního kanálu s kódovacím zařízením zprávy

Vytvořte vlastní vazbu přidáním následujícího kódu do vaší aplikace:

```
Binding binding = new CustomBinding(new TextMessageEncodingBindingElement(),
transportBindingElement);
```

Krok 4: Vytvoření identifikátoru URI SOAP/JMS

Jako adresu koncového bodu je třeba zadat identifikátor URI SOAP/JMS, který popisuje podrobnosti připojení produktu WebSphere MQ vyžadované k přístupu ke službě. Závisí to na tom, zda je kanál používán pro aplikaci služby nebo pro klientskou aplikaci.

Pro klientské aplikace musí být identifikátor URI SOAP/JMS vytvořen jako `EndpointAddress` takto:

```
EndpointAddress address = new EndpointAddress("jms:/queue?
destination=SampleQ@QM1&connectionFactory=connectQueueManager(QM1)&initialContextFactory=com.ibm
.mq.jms.Nojndi");
```

Pro servisní aplikace musí být identifikátor URI SOAP/JMS vytvořen následujícím způsobem:

```
Uri address = new Uri("jms:/queue?
destination=SampleQ@QM1&connectionFactory=connectQueueManager(QM1)&initialContextFactory=com.ibm
.mq.jms.Nojndi");
```

Další informace o adrese koncového bodu naleznete v tématu: “Vlastní kanál produktu WebSphere MQ pro formát adresy identifikátoru URI koncového bodu služby WCF” na stránce 587

Vlastní kanál produktu WebSphere MQ pro formát adresy identifikátoru URI koncového bodu služby WCF

Identifikátor URI (Universal Resource Identifier) poskytuje podrobnosti o umístění a připojení pro určení webové služby. Tento formát identifikátoru URI povoluje během přístupu k cílovým službám komplexní stupeň řízení pro parametry a volby specifické pro produkt SOAP/ WebSphere MQ.

Webová služba je uvedena pomocí identifikátoru URI (Universal Resource Identifier). Tato sekce uvádí formát identifikátoru URI, který je podporován v přenosu WebSphere MQ pro SOAP. Tento formát

identifikátoru URI povoluje během přístupu k cílovým službám komplexní stupeň řízení pro parametry a volby specifické pro produkt SOAP/WebSphere MQ. Tento formát je kompatibilní s produktem WebSphere Application Server (WAS) a s produktem CICS usnadněním integrace produktu WebSphere MQ s oběma těmito produkty.

Syntaxe identifikátoru URI je následující:

```
jms:/queue?name=value&name=value...
```

kde name je název parametru a *hodnota* je vhodnou hodnotou a prvek name=*value* lze opakovat vícekrát s druhým a následnými výskyty, před nimiž je uveden ampersand (&).

Další informace o nastavení vlastností identifikátoru URI viz: Syntaxe identifikátoru URI a parametry pro implementaci webové služby

Názvy parametrů rozlišují velikost písmen, jako jsou názvy objektů WebSphere MQ . Je-li některý parametr zadán vícekrát než jednou, bude konečným výskytem tohoto parametru účinek, což znamená, že klientské aplikace mohou hodnoty parametrů přepsat připojením k identifikátoru URI. Jsou-li zahrnuty nějaké další nerozpoznané parametry, jsou ignorovány.

Pokud ukládáte identifikátor URI do řetězce XML, musíte znázornit znak ampersand ve tvaru "&". Podobně platí, že pokud je identifikátor URI kódován ve skriptu, postarejte se o řídicí znaky, jako je **&** , které by jinak shell interpretoval.

Toto je příklad jednoduchého identifikátoru URI pro službu Axis:

```
jms:/queue?destination=myQ&connectionFactory=()
&initialContextFactory=com.ibm.mq.jms.Nojndi
```

Zde je příklad jednoduchého identifikátoru URI pro službu .NET:

```
jms:/queue?destination=myQ&connectionFactory=()&targetService=MyService.asmx
&initialContextFactory=com.ibm.mq.jms.Nojndi
```

Dodané jsou pouze požadované parametry (`targetService` je povinné pouze pro služby .NET) a `connectionFactory` nejsou žádné volby.

V tomto příkladu osy `connectionFactory` obsahuje několik 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 zadána také volba `sslPeerName` parametru `connectionFactory` . Hodnota samotného názvu sslPeerobsahuje dvojice název-hodnota a významné vložené mezery:

```
jms:/queue?destination=myQ@myRQM&connectionFactory=connectQueueManager(myconnQM)
binding(client)clientChannel(myChannel)clientConnection(myConnection)
sslPeerName(CN=MQ Test 1,O=IBM,S=Hampshire,C=GB)
&initialContextFactory=com.ibm.mq.jms.Nojndi
```

Volby konfigurace vazby WCF

Toto téma popisuje, jak lze použít volby konfigurace na informace o vazbě vlastního kanálu a zobrazit volby, které jsou k dispozici.

Volby konfigurace vazby lze nastavit jedním ze dvou způsobů:

1. Administrativně: Nastavení vlastností vazby musí být určena v části transportu definice vlastní vazby v konfiguračním souboru aplikací, například: `app.config`
2. Programově: Kód aplikace musí být upraven tak, aby určoval vlastnost při inicializaci vlastní vazby.

Administrativně nastavení vlastností vázání
Nastavení vlastností vazby lze také zadat v konfiguračním souboru aplikace, například: `app.config`. Konfigurační soubor je generován produktem **svcutil**, například:

```
<customBinding>

...

     <IBM.XMS.WCF.SoapJmsIbmTransportChannel maxBufferPoolSize="524288"
      maxMessageSize="4000000" clientConnectionMode="0" maxConcurrentCalls="16"/>

...

</customBinding>
```

Programové nastavení vlastností vazby

Chcete-li přidat vlastnost vazby WCF k určení režimu připojení klienta, je třeba upravit kód služby tak, aby určoval vlastnost během inicializace vlastní vazby.

Při zadávání režimu připojení nespravovaného klienta použijte následující příklad:

```
SoapJmsIbmTransportBindingElement transportBindingElement = new
SoapJmsIbmTransportBindingElement();
transportBindingElement.ClientConnectionMode = XmsWCFBindingProperty.CLIENT_UNMANAGED;

Binding sampleBinding = new CustomBinding(new TextMessageEncodingBindingElement(),
                                                    transportBindingElement);
```

Vlastnosti vazby WCF

Název vlastnosti	Aplikace klienta nebo služby	Administrativní hodnota	Programová hodnota	Popis
maxBufferPoolSize	Oboje	0 až 64 bitů podepsané celé číslo	0 až 64 bitů podepsané celé číslo	Určuje maximální velikost paměti, kterou lze použít k uložení vyrovnávacích pamětí zpráv WCF pro instanci kanálu.
Velikost maxMessage	Oboje	1 až 32 bitů signed integer	1 až 32 bitů signed integer	Určuje maximální velikost paměti, kterou lze použít pro jednotlivé zprávy WCF.
Režim clientConnection	Oboje	0 (Výchozí hodnota) 1	AS_URI (Výchozí hodnota) NESPRAVOVÁNO KLIENTA	Určuje režim připojení klienta transportního kanálu. Hodnota 0 znamená, že režim připojení klienta je stejný jako v identifikátoru URI. Použije se pouze v případě, že je použito připojení klienta. Určuje, že režim připojení klienta je určen v identifikátoru URI. Hodnota 0 je výchozí hodnotou, pokud není nastaven režim připojení klienta. Hodnota 1 znamená, že režim připojení klienta je nespravovaný klient. Použije se pouze v případě, že je použito připojení klienta.

Název vlastnosti	Aplikace klienta nebo služby	Administrativní hodnota	Programová hodnota	Popis
Volání MaxConcurrent	Klient	Rozsah je 0-2 147 483 647 16 je výchozí hodnota	Rozsah je 0-2 147 483 647 16 je výchozí hodnota	Tato vlastnost definuje maximální počet souběžných operací, které mohou být prováděny na individuálním serveru proxy klienta najednou. Je-li spuštěno více operací, jsou řazeny do fronty, dokud nebude dokončena nebo ukončena operace probíhající nebo probíhající. Toto nastavení lze použít k řízení maximálního počtu podprocesů a prostředků, které může spotřebovávat jednotlivý server proxy. Hodnota `0` tento limit odebere, a umožní tak pokus o provedení všech operací současně.
Volání MaxConcurrent	Služba	Rozsah je 1-2 147 483 647 16 je výchozí hodnota	Rozsah je 1-2 147 483 647 16 je výchozí hodnota	Tato vlastnost se používá pouze v případě, že je povolena funkce zajištění doručení (Další informace o zajištěného doručení viz “Doručení vlastního kanálu WCF” na stránce 580). Určuje maximální počet souběžných operací, které mohou ve stejnou dobu pro daný koncový bod probíhat současně. Při změně tohoto nastavení je třeba věnovat pozornost. Každá souběžná operace vyžaduje další prostředky, zejména novou instanci vlastního kanálu a přidružené podprocesy z fondu podprocesů, aby bylo možné provést příslušné akce. Nadalokace může být kontraproduktivní a nepříznivě ovlivnit výkonnost. Aby bylo možné tuto vlastnost podporovat, musí být vytvořena odpovídající konfigurace fondu podprocesů.

Budování a hosting služeb pro WCF

Overview of Microsoft Windows Communication Foundation (WCF) services explaining how to create and configure WCF services.

Vlastní kanál produktu IBM WebSphere MQ pro službu WCF a služby WCF, které jej používají, může být hostován následujícími způsoby:

- Samohostitelství

- Služba systému Windows

Vlastní kanál produktu IBM WebSphere MQ pro službu WCF nemůže být hostován ve službě Windows Process Activation Service.

Následující témata poskytují některé jednoduché příklady samohostování k demonstraci příslušných kroků. Online dokumentace 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: vlastní hostování administrativně pomocí konfiguračního souboru aplikace

Poté, co jste vytvořili konfigurační soubor aplikace, otevřete instanci služby a přidejte do aplikace uvedený kód.

Než začnete

Vytvořte nebo upravte konfigurační soubor aplikace pro službu, jak je popsáno v: "Vytvoření vlastního uživatelského kanálu WCF administrativně dodáním informací o vazbách a koncových bodech v konfiguračním souboru aplikace" na stránce 584

Informace o této úloze

1. Vytvořit instanci a otevřít instanci služby v hostiteli služby. Typ služby musí být stejný jako typ služby uvedený v konfiguračním souboru služby.
2. Přidejte do své aplikace tento kód:

```
ServiceHost service = new ServiceHost(typeof(MyService));
service.Open();
...
service.Close();
```

Sestavování aplikací služeb WCF pomocí metody 2: Samoobsluha programově přímo z aplikace

Přidejte vlastnosti vazby, vytvořte hostitele služby s instancí požadované třídy služeb a otevřete danou službu.

Než začnete

1. Přidejte odkaz na soubor vlastního kanálu `IBM.XMS.WCF.dll` do projektu. Produkt `IBM.XMS.WCF.dll` se nachází v adresáři *`WMQInstallDir`*`\bin` , kde *WMQInstallDir* je adresář, ve kterém je nainstalován produkt WebSphere MQ 7.
2. Přidejte příkaz *using* do oboru názvů `IBM.XMS.WCF` , například: `using IBM.XMS.WCF`
3. Vytvořte instanci prvku vazby kanálů a koncového bodu, jak je popsáno v: "Vytvoření vlastního kanálu WCF nahrazením informací o vazbách a koncovém bodu programově" na stránce 586

Informace o této úloze

Jsou-li vyžadovány změny vlastností vazby kanálu, postupujte takto:

1. Přidejte vlastnosti vazby do produktu `transportBindingElement` , jak ukazuje následující příklad:

```
SoapJmsIbmTransportBindingElement transportBindingElement = new
SoapJmsIbmTransportBindingElement();
Binding binding = new CustomBinding(new TextMessageEncodingBindingElement(),
transportBindingElement);
Uri address = new Uri("jms:/queue?destination=SampleQ@QM1&connectionFactory=

connectQueueManager(QM1)&initialContextFactory=com.ibm.mq.jms.Nojndi");
```

2. Vytvořte hostitele služby s instancí požadované třídy služeb:

```
ServiceHost service = new ServiceHost(typeof(MyService));
```

3. Otevřete službu:

```
service.AddServiceEndpoint(typeof(IMyServiceContract), binding, address);
service.Open();
    ...
service.Close();
```

Vystavení metadat pomocí koncového bodu HTTP

Pokyny pro vystavení metadat služby, která je nakonfigurována pro použití vlastního kanálu produktu WebSphere MQ pro produkt WCF.

Informace o této úloze

Pokud metadata služeb musí být odkryta (aby mohly být nástroje, jako například produkt `svcutil` , k němu přistupovat přímo ze spuštěné služby spíše než ze souboru WSDL offline), musí být provedeny vystavením metadat služeb koncovým bodem protokolu HTTP. Následující kroky lze použít k přidání tohoto dalšího koncového bodu.

1. Přidejte základní adresu, kde musí být metadata vystavena objektu ServiceHost, například:

```
ServiceHost service = new ServiceHost(typeof(TestService),
                      new Uri("http://localhost:8000/MyService"));
```

2. Před otevřením služby přidejte následující kód do ServiceHost :

```
ServiceMetadataBehavior metadataBehavior = new ServiceMetadataBehavior();
  metadataBehavior.HttpGetEnabled = true;
  service.Description.Behaviors.Add(metadataBehavior);
  service.AddServiceEndpoint(typeof(IMetadataExchange),
     MetadataExchangeBindings.CreateMexHttpBinding(), "mex");
```

Výsledky

Metadata jsou nyní k dispozici na této adrese: `http://localhost:8000/MyService`

Sestavování aplikací klienta pro WCF

Přehled generování a sestavení klientských aplikací systému Microsoft Windows Communication Foundation (WCF).

Klientská aplikace může být vytvořena pro službu WCF; klientské aplikace se obvykle generují pomocí obslužného nástroje Microsoft ServiceModel Metadata Utility Tool (Svcutil.exe) k vytvoření požadované konfigurace a souborů serveru proxy, které mohou být použity přímo aplikací.

Generování serveru proxy klienta WCF a konfiguračních souborů aplikace pomocí nástroje svcutil s metadaty ze spuštěné služby

Pokyny pro použití nástroje Microsoft svcutil.exe ke generování klienta pro službu, která je nakonfigurována pro použití vlastního kanálu produktu WebSphere MQ pro WCF.

Než začnete

K dispozici jsou tři předpoklady pro použití nástroje svcutil k vytvoření požadované konfigurace a souborů proxy, které lze použít přímo aplikací:

- Služba WCF musí být spuštěna dříve, než se spustí nástroj svcutil.
- Služba WCF musí vystavit svá metadata pomocí portu HTTP spolu s odkazy na koncový bod vlastního kanálu produktu WebSphere MQ tak, aby generoval klienta přímo ze spuštěné služby.
- Vlastní kanál musí být registrován v konfiguračních datech pro svcutil.

Informace o této úloze

Následující kroky vysvětlují, jak generovat klienta pro službu, která je nakonfigurována pro použití vlastního kanálu produktu WebSphere MQ , ale také odkrývá svá metadata za běhu prostřednictvím samostatného portu HTTP:

1. Spusťte službu WCF (služba musí být spuštěna dříve, než se spustí nástroj svcutil).
2. Přidejte podrobnosti z konfiguračního souboru `svcutil.exe` z kořenového adresáře instalace do aktivního konfiguračního souboru svcutil, zpravidla `C:\Program Files\Microsoft SDKs\Windows\v6.0A\bin\svcutil.exe.config` , takže svcutil rozpozná vlastní kanál produktu WebSphere 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 Microsoft Visual studio.

Jak pokračovat dále

Pokud nelze metadata služeb přímo načíst, lze místo toho použít svcutil ke generování souborů klienta z wsdl. Další informace viz: “Generování serveru proxy klienta WCF a konfiguračních souborů aplikace pomocí nástroje svcutil s kódem WSDL” na stránce 593

Generování serveru proxy klienta WCF a konfiguračních souborů aplikace pomocí nástroje svcutil s kódem WSDL

Pokyny pro generování klientů WCF z WSDL, pokud metadata služby nejsou k dispozici.

Pokud metadata služby nemohou být přímo načtena pro generování klienta z metadat ze spuštěné služby, pak lze použít svcutil k vygenerování klientských souborů z WSDL. Chcete-li určit, že má být použit vlastní kanál produktu WebSphere MQ , musí být provedeny následující úpravy souboru WSDL:

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:ExactlyOne>
                <wsp:All>
                    <xms:xms xmlns:xms="http://sample.schemas.ibm.com/policy/xms" />
                </wsp:All>
            </wsp:ExactlyOne>
        </wsp:Policy>

...

</wsdl:definitions>
```

2. Upravte sekci vazeb tak, aby odkazovala na novou sekci zásady, a odeberte definici produktu `transport` ze základního prvku vazby:

```
<wsdl:definitions ...>

        <wsdl:binding ...>
                <wsp:PolicyReference URI="#CustomerBinding_IWMQSampleContract_policy" />
                <[soap]:binding ... transport="" />
                ...
        </wsdl:binding>
</wsdl:definitions>
```

3. Spusťte svcutil z příkazového řádku, například:

```
svcutil /language:C# /r:MQ_INSTALLATION_PATH\bin\IBM.XMS.WCF.dll
        /config:app.config
MQ_INSTALLATION_PATH\src\samples\WMQAxis\default\service\soap.server.stockQuoteAxis_Wmq.wsdl
```

Kde *MQ_INSTALLATION_PATH* je instalační adresář produktu WebSphere MQ.

Sestavování aplikací klienta WCF pomocí serveru proxy klienta s konfiguračním souborem aplikace

Než začnete

Vytvořte nebo upravte konfigurační soubor aplikace pro klienta, jak je popsáno v tématu: “Vytvoření vlastního uživatelského kanálu WCF administrativně dodáním informací o vazbách a koncových bodech v konfiguračním souboru aplikace” na stránce 584

Informace o této úloze

Vytvořit instanci a otevřít instanci serveru proxy klienta. Parametr předaný generovanému serveru proxy musí být stejný jako název koncového bodu určený v konfiguračním souboru klienta, například `Endpoint_WMQ`:

```
MyClientProxy myClient = new MyClientProxy("Endpoint_WMQ");
            try {
                   myClient.myMethod("HelloWorld!");
                   myClient.Close();
            }
            catch (TimeoutException e) {
                Console.Out.WriteLine(e);
                myClient.Abort();
            }
            catch (CommunicationException e) {
                Console.Out.WriteLine(e);
                myClient.Abort();
            }
            catch (Exception e) {
                Console.Out.WriteLine(e);
                myClient.Abort();
            }
```

Sestavování aplikací klienta WCF s použitím serveru proxy klienta s programovou konfigurací

Než začnete

1. Přidejte odkaz na soubor vlastního kanálu `IBM.XMS.WCF.dll` do projektu. `IBM.XMS.WCF.dll` je v adresáři *WMQInstallDir*`\bin` , kde *WMQInstallDir* je adresář, ve kterém je nainstalován produkt WebSphere MQ 7.
2. Přidejte příkaz *using* do oboru názvů `IBM.XMS.WCF` , například: `using IBM.XMS.WCF`
3. Vytvořte instanci prvku vazby a koncový bod kanálu, jak je popsáno v tématu: “Vytvoření vlastního kanálu WCF nahrazením informací o vazbách a koncovém bodu programově” na stránce 586

Informace o této úloze

Jsou-li vyžadovány změny vlastností vazby kanálu, postupujte takto:

1. Přidejte vlastnosti vazby do produktu `transportBindingElement` , jak ukazuje následující obrázek:

```
SoapJmsIbmTransportBindingElement transportBindingElement = new
SoapJmsIbmTransportBindingElement();
Binding binding = new CustomBinding(new TextMessageEncodingBindingElement(),
transportBindingElement);
EndpointAddress address =
  new EndpointAddress("jms:/queue?destination=SampleQ@QM1&connectionFactory=

connectQueueManager(QM1)&initialContextFactory=com.ibm.mq.jms.Nojndi");
```

2. Vytvořte server proxy klienta, jak ukazuje následující obrázek, kde *binding* a *endpoint address* jsou vazba a adresa koncového bodu konfigurované v kroku “1” na stránce 594 a předány v:

```
MyClientProxy myClient = new MyClientProxy(binding, endpoint address);
           try {
                  myClient.myMethod("HelloWorld!");
                  myClient.Close();
           }
           catch (TimeoutException e) {
               Console.Out.WriteLine(e);
               myClient.Abort();
           }
           catch (CommunicationException e) {
               Console.Out.WriteLine(e);
                               myClient.Abort();
           }
           catch (Exception e) {
               Console.Out.WriteLine(e);
               myClient.Abort();
           }
```

Použití ukázek WCF

Ukázky produktu Okna Communication Foundation (WCF) poskytují některé jednoduché příklady použití vlastního kanálu produktu WebSphere MQ .

Chcete-li sestavit ukázkové projekty, potřebujete buď sadu Microsoft .NET 3.5 SDK, nebo produkt Microsoft Visual Studio 2008.

Ukázka klienta WCF jednosměrného klienta a serveru

Tato ukázka předvádí vlastní kanál produktu WebSphere MQ , který se používá ke spuštění služby WCF (Windows Communication Foundation) z klienta WCF pomocí tvaru jednosměrného kanálu.

Informace o této úloze

Služba implementuje jedinou metodu, jejíž výstupem je řetězec na konzolu. Klient byl generován pomocí nástroje `svcutil` k načtení metadat služby ze samostatně vystaveného koncového bodu HTTP, jak je popsáno v tématu “Generování serveru proxy klienta WCF a konfiguračních souborů aplikace pomocí nástroje svcutil s metadaty ze spuštěné služby” na stránce 592 .

Ukázka byla nakonfigurována se specifickými názvy prostředků, jak je popsáno v následujícím postupu. Musíte-li změnit názvy prostředků, musíte také změnit odpovídající hodnotu v klientské aplikaci v souboru *MQ_INSTALLATION_PATH*`\tools\dotnet\samples\cs\wcf\samples\WCF\oneway\client\app.config` a v aplikaci služby 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 WebSphere MQ. Další informace o formátování identifikátoru URI koncového bodu rozhraní JMS naleznete v tématu *WebSphere MQ Transport pro SOAP* v dokumentaci produktu WebSphere MQ . Potřebujete-li upravit ukázkové řešení a zdroje, pak potřebujete prostředí IDE, například Microsoft Visual Studio 8 nebo vyšší.

Postup

1. Vytvořte správce front s názvem *QM1* .
2. Vytvořte místo určení fronty s názvem *SampleQ* .
3. Spusťte službu tak, aby modul listener čekal na zprávy: Spusťte soubor *MQ_INSTALLATION_PATH*`\tools\dotnet\samples\cs\wcf\samples\WCF\oneway\service\bin\Release\TestService.exe` , kde *MQ_INSTALLATION_PATH* je instalační adresář pro IBM WebSphere 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 WebSphere MQ.

Klientská aplikační smyčka pětkrát odesílá pět zpráv do *SampleQ* .

Výsledky

Aplikace služeb získává zprávy z *SampleQ* a na obrazovce se zobrazí `Hello World` pětkrát.

Jak pokračovat dále

Jednoduchý požadavek-klient odpovědí a server WCF serveru

Tato ukázka předvádí vlastní kanál produktu WebSphere MQ , který se používá ke spuštění služby WCF (Windows Communication Foundation) z klienta WCF pomocí tvaru kanálu s odpovědí typu požadavek-odezva.

Informace o této úloze

Tato služba poskytuje některé jednoduché metody kalkulačky k přidání a odečtení dvou čísel a následné vrácení výsledku. Klient byl generován pomocí nástroje `svcutil` k načtení metadat služby ze samostatně vystaveného koncového bodu HTTP, jak je popsáno v tématu "Generování serveru proxy klienta WCF a konfiguračních souborů aplikace pomocí nástroje svcutil s metadaty ze spuštěné služby" na stránce 592 .

Ukázka byla nakonfigurována se specifickými názvy prostředků, jak je popsáno v následujícím postupu. Potřebujete-li změnit názvy prostředků, musíte také změnit odpovídající hodnotu v klientské aplikaci v souboru *MQ_INSTALLATION_PATH*`\Tools\wcf\samples\WCF\requestreply\client\app.config` a v aplikaci služeb v souboru *MQ_INSTALLATION_PATH*`\Tools\wcf\samples\WCF\requestreply\service\RequestReplyService.cs` , kde *MQ_INSTALLATION_PATH* je instalační adresář pro produkt WebSphere MQ. Další informace o formátování identifikátoru URI koncového bodu rozhraní JMS naleznete v tématu *WebSphere MQ Transport pro SOAP* v dokumentaci produktu WebSphere MQ . Potřebujete-li upravit ukázkové řešení a zdroje, pak potřebujete prostředí IDE, například Microsoft Visual Studio 8 nebo vyšší.

Postup

1. Vytvořte správce front s názvem *QM1* .
2. Vytvořte místo určení fronty s názvem *SampleQ* .
3. Vytvořte cíl fronty s názvem *SampleReplyQ* .
4. Spusťte službu tak, aby modul listener čekal na zprávy: Spusťte soubor *MQ_INSTALLATION_PATH*`\Tools\wcf\samples\WCF\requestreply\service\bin\Release\SimpleRequestReply_Service.exe` , kde *MQ_INSTALLATION_PATH* je instalační adresář produktu WebSphere 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 produktu WebSphere MQ.

Výsledky

Když je klient spuštěn, spustí se následující proces a opakuje se čtyřikrát, takže se každý z pěti zpráv posílá každý:

1. Klient vloží zprávu s požadavkem na adresu *SampleQ* a čeká na odpověď.
2. Služba získá zprávu požadavku z adresáře *SampleQ*.
3. Služba přidá a odečítá některé hodnoty pomocí obsahu zprávy.
4. Služba pak vloží výsledky do zprávy v *SampleReplyQ*a čeká, až klient vloží novou zprávu.
5. Klient získá zprávu z fronty *SampleReplyQ* a zobrazí výsledky na obrazovce.

Jak pokračovat dále

Klient WCF, který je hostitelem služby .NET, jehož hostitelem je produkt WebSphere MQ

Ukázkové klientské aplikace a ukázkové aplikace serveru proxy služeb jsou dodávány pro prostředí .NET i Java. Ukázky jsou založeny na službě Stock Quote, která přijímá požadavek na akcie akcií a poté poskytuje kótování akcií.

Než začnete

Ukázka vyžaduje správnou instalaci a konfiguraci běhového prostředí služby .NET SOAP over JMS v produktu WebSphere MQ a je přístupný z lokálního správce front. Chcete-li získat informace o instalaci a konfiguraci prostředí, prohlédněte si: “Instalace produktu WebSphere MQ Web transport pro SOAP” na stránce 924

Je-li prostředí .NET SOAP prostřednictvím služby JMS správně nainstalováno a nakonfigurováno v produktu WebSphere MQ a je přístupné z lokálního správce front, je třeba provést další kroky konfigurace.

1. Nastavte proměnnou prostředí WMQSOAP_HOME na instalační adresář produktu WebSphere MQ , například: `C:\Program Files\IBM\WebSphere MQ`
2. Ujistěte se, že kompilátor jazyka Java `javac` je dostupný a na cestě PATH.
3. Zkopírujte soubor `axis.jar` z adresáře `prereqs/axis` instalačního disku CD WebSphere do produkčního adresáře produktu WebSphere MQ , například: `C:\Program Files\IBM\WebSphere MQ\java\lib\soap`
4. Přidejte do proměnné PATH: *MQ_INSTALLATION_PATH*`\Java\lib` , kde *MQ_INSTALLATION_PATH* představuje adresář, do kterého je nainstalován produkt WebSphere MQ , například: `C:\Program Files\IBM\WebSphere MQ`
5. Ujistěte se, že umístění prostředí .NET je v produktu *MQ_INSTALLATION_PATH*`\bin\amqwcallWSDL.cmd` , kde *MQ_INSTALLATION_PATH* představuje adresář, kde je instalován produkt WebSphere MQ , například: `C:\Program Files\IBM\WebSphere MQ`, je správně určen. Umístění platformy .NET může být zadáno například: `set msfwdir=%ProgramFiles%\Microsoft Visual Studio .NET 2003\SDK\v1.1\Bin`

Když jsou předchozí kroky dokončeny, otestujte a spusťte službu:

1. Přejděte do pracovního adresáře protokolu SOAP prostřednictvím rozhraní JMS.
2. Zadejte jeden z následujících příkazů pro spuštění testu ověření a nechte spuštěný modul listener služby:
   - Pro .NET: *MQ_INSTALLATION_PATH*`\Tools\soap\samples\runivt dotnet hold` , kde *MQ_INSTALLATION_PATH* představuje adresář, kde je nainstalován produkt WebSphere MQ .
   - Pro AXIS:*MQ_INSTALLATION_PATH*`\Tools\soap\samples\runivt Dotnet2AxisClient hold` kde *MQ_INSTALLATION_PATH* představuje adresář, kde je nainstalován produkt WebSphere MQ .

Argument `hold` uchovává moduly listener spuštěné po dokončení testu.

Pokud jsou během této konfigurace ohlášeny chyby, můžete odebrat všechny změny, aby bylo možné proceduru restartovat následujícím způsobem:

1. Odstraňte vygenerovaný adresář protokolu SOAP prostřednictvím rozhraní JMS.
2. Odstraňte správce front.

Informace o této úloze

Tato ukázka předvádí připojení z klienta WCF k ukázkové službě rozhraní .NET SOAP prostřednictvím rozhraní JMS poskytované v produktu WebSphere MQ pomocí jednosměrného tvaru kanálu. Služba implementuje jednoduchý příklad StockQuote , který výstupem je textový řetězec na konzolu.

Klient byl vygenerován pomocí jazyka WSDL ke generování souborů klienta, jak je popsáno v tématu "Generování serveru proxy klienta WCF a konfiguračních souborů aplikace pomocí nástroje svcutil s kódem WSDL" na stránce 593 .

Ukázka byla nakonfigurována se specifickými názvy prostředků, jak je popsáno v následujícím postupu. Potřebujete-li změnit názvy prostředků, musíte také změnit odpovídající hodnotu v klientské aplikaci v souboru *MQ_INSTALLATION_PATH*\tools\wcf\samples\WMQNET\default\client\app.config a na aplikaci služby v souboru *MQ_INSTALLATION_PATH*\tools\wcf\samples\WMQNET\default\service\WmqDefaultSample_StockQuoteDotNet.wsdl , kde *MQ_INSTALLATION_PATH* představuje instalační adresář pro produkt WebSphere MQ. Další informace o formátování identifikátoru URI koncového bodu rozhraní JMS naleznete v tématu *WebSphere MQ Transport pro SOAP* v dokumentaci produktu WebSphere 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 produkt WebSphere MQ.

Klientská aplikační smyčka pětkrát odesílá pět zpráv do ukázkové fronty.

Výsledky

Aplikace služby získá zprávy z ukázkové fronty a zobrazí `Hello World` pětkrát na obrazovce.

Klient WCF do služby Axis Java hostované ukázkou produktu WebSphere MQ

Ukázkové klientské aplikace a ukázkové aplikace serveru proxy služeb jsou dodávány pro prostředí Java i .NET. Ukázky jsou založeny na službě Stock Quote, která přijímá požadavek na akcie akcií a poté poskytuje kótování akcií.

Než začnete

Tato ukázka vyžaduje, aby prostředí hostitele služby .NET SOAP over JMS bylo správně nainstalováno a nakonfigurováno v produktu WebSphere MQ a že je přístupné z lokálního správce front. Chcete-li získat informace o instalaci a konfiguraci prostředí, prohlédněte si: "Instalace produktu WebSphere MQ Web transport pro SOAP" na stránce 924

Je-li prostředí .NET SOAP prostřednictvím služby JMS správně nainstalováno a nakonfigurováno v produktu WebSphere MQ a je přístupné z lokálního správce front, je třeba provést další kroky konfigurace.

1. Nastavte proměnnou prostředí WMQSOAP_HOME na instalační adresář produktu WebSphere MQ , například: `C:\Program Files\IBM\WebSphere MQ`
2. Ujistěte se, že kompilátor jazyka Java `javac` je dostupný a na cestě PATH.
3. Zkopírujte soubor `axis.jar` z adresáře `prereqs/axis` instalačního disku CD WebSphere do instalačního adresáře produktu WebSphere MQ .
4. Přidejte do proměnné PATH: *MQ_INSTALLATION_PATH*\Java\lib , kde *MQ_INSTALLATION_PATH* představuje adresář, do kterého je nainstalován produkt WebSphere MQ , například: `C:\Program Files\IBM\WebSphere MQ`
5. Ujistěte se, že umístění prostředí .NET je v produktu *MQ_INSTALLATION_PATH*\bin\amqwcallWSDL.cmd , kde *MQ_INSTALLATION_PATH* představuje adresář, kde je instalován produkt WebSphere MQ , například: `C:\Program Files\IBM\WebSphere MQ`, je správně určen. Umístění platformy .NET může být zadáno například: `set msfwdir=%ProgramFiles%\Microsoft Visual Studio .NET 2003\SDK\v1.1\Bin`

Když jsou předchozí kroky dokončeny, otestujte a spusťte službu:

1. Přejděte do pracovního adresáře protokolu SOAP prostřednictvím rozhraní JMS.
2. Zadejte jeden z následujících příkazů pro spuštění testu ověření a nechte spuštěný modul listener služby:

- Pro .NET: *MQ_INSTALLATION_PATH*\Tools\soap\samples\runivt dotnet hold , kde *MQ_INSTALLATION_PATH* představuje adresář, kde je nainstalován produkt WebSphere MQ .
- Pro AXIS:*MQ_INSTALLATION_PATH*\Tools\soap\samples\runivt Dotnet2AxisClient hold kde *MQ_INSTALLATION_PATH* představuje adresář, kde je nainstalován produkt WebSphere MQ .

Argument hold uchovává moduly listener spuštěné po dokončení testu.

Pokud jsou během této konfigurace ohlášeny chyby, můžete odebrat všechny změny tak, aby se procedura restartovala následujícím způsobem:

1. Odstraňte vygenerovaný adresář protokolu SOAP prostřednictvím rozhraní JMS.
2. Odstraňte správce front.

Informace o této úloze

Ukázka demonstruje připojení z klienta WCF k ukázkové službě SOAP Java SOAP prostřednictvím rozhraní JMS poskytované v produktu WebSphere MQ pomocí tvaru jednosměrného kanálu. Služba implementuje jednoduchý příklad StockQuote , který posílá textový řetězec do souboru, který je uložen v aktuálním adresáři.

Klient byl vygenerován pomocí jazyka WSDL ke generování souborů klienta, jak je popsáno v tématu “Generování serveru proxy klienta WCF a konfiguračních souborů aplikace pomocí nástroje svcutil s kódem WSDL” na stránce 593 .

Ukázka byla nakonfigurována se specifickými názvy prostředků, jak je popsáno v tomto odstavci. Potřebujete-li změnit názvy prostředků, musíte také změnit odpovídající hodnotu v klientské aplikaci v souboru *MQ_INSTALLATION_PATH*\tools\wcf\samples\WMQAxis\default\client\app.config a na aplikaci služby v souboru *MQ_INSTALLATION_PATH*\tools\wcf\samples\WMQAxis\default\service\WmqDefaultSample_StockQuoteDotNet.wsdl , kde *MQ_INSTALLATION_PATH* představuje instalační adresář pro produkt WebSphere 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 produkt WebSphere MQ.

Klientská aplikační smyčka pětkrát odesílá pět zpráv do ukázkové fronty.

Výsledky

Aplikace služby získá zprávy z ukázkové fronty a přidá Hello World pětkrát do souboru v aktuálním adresáři.

Související odkazy

“Ošetřování různých názvů prvků odezvy SOAP” na stránce 607

WCF očekává, že název navrácené hodnoty bude standardně ve specifickém formátu, ale služba nemusí vrátit prvek s jeho jménem v očekávaném formátu.

Klient WCF do služby Java, jehož hostitelem je ukázka serveru WebSphere Application Server

Ukázkové klientské aplikace a ukázkové aplikace proxy služeb jsou dodávány pro produkt WebSphere Application Server (WAS) 6. Poskytne se také služba požadavek-odezva.

Než začnete

Tato ukázka vyžaduje použití následující konfigurace produktu WebSphere MQ :

Tabulka 77. Požadovaná konfigurace produktu WebSphere MQ	
Objekt	**Povinné jméno**
Správce front	QM1
Lokální fronta	HelloWorld
Lokální fronta	Odpověď HelloWorld

Tato ukázka také vyžaduje, aby bylo správně nainstalováno a nakonfigurováno hostitelské prostředí serveru WebSphere Application Server V6 . Produkt WebSphere Application Server V6 používá při výchozím nastavení připojení režimu vazeb k produktu WebSphere MQ . Proto musí být server WebSphere Application Server V6 nainstalován na stejném počítači jako správce front.

Po konfiguraci prostředí WAS musí být dokončeny následující další kroky konfigurace:

1. V úložišti JNDI serveru WebSphere Application Server vytvořte následující objekty JNDI:
   a. Místo určení fronty JMS s názvem `HelloWorld`
      - Nastavte název rozhraní JNDI na hodnotu `jms/HelloWorld` .
      - Nastavit název fronty na `HelloWorld`
   b. Továrna připojení fronty JMS s názvem `HelloWorldQCF`
      - Nastavte název rozhraní JNDI na hodnotu `jms/HelloWorldQCF` .
      - Nastavit název správce front na `QM1`
   c. Továrna připojení fronty JMS s názvem `WebServicesReplyQCF`
      - Nastavte název rozhraní JNDI na hodnotu `jms/WebServicesReplyQCF` .
      - Nastavit název správce front na `QM1`
2. Vytvořte port modulu listener pro zprávy s názvem `HelloWorldPort` na serveru WebSphere Application Server s následující konfigurací:
   - Nastavte název rozhraní JNDI továrny připojení na `jms/HelloWorldQCF`
   - Nastavte název rozhraní JNDI místa určení na hodnotu `jms/HelloWorld` .
3. Nainstalujte aplikaci webové služby `HelloWorldEJBEAR.ear` na server WebSphere Application Server tímto 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 WebSphere MQ.
   c. Neměňte žádnou výchozí volbu v průvodci a restartujte aplikační server po instalaci aplikace.

Po dokončení konfigurace serveru WAS ji otestujte spuštěním této služby:

1. Přejděte na pracovní adresář služby Soap over JMS.
2. Zadejte tento příkaz ke spuštění ukázky: *MQ_INSTALLATION_PATH*`\tools\wcf\samples\WAS\TestClient.exe` , kde *MQ_INSTALLATION_PATH* je instalační adresář produktu WebSphere MQ.

Informace o této úloze

Ukázka demonstruje připojení z klienta WCF na ukázkovou službu WebSphere Application Server SOAP over JMS, která je součástí ukázek WCF zahrnutých v produktu WebSphere MQ V7pomocí tvaru kanálu požadavek-odezva. Tok zpráv mezi WCF a serverem WebSphere Application Server pomocí front produktu WebSphere MQ . Služba implementuje metodu `HelloWorld(...)` , která přijímá řetězec a vrací pozdrav klientovi.

Klient byl generován pomocí nástroje svcutil k načtení metadat služby ze samostatně vystaveného koncového bodu HTTP, jak je popsáno v tématu “Generování serveru proxy klienta WCF a konfiguračních souborů aplikace pomocí nástroje svcutil s metadaty ze spuštěné služby” na stránce 592

Ukázka byla nakonfigurována se specifickými názvy prostředků, jak je popsáno v následujícím postupu. Potřebujete-li změnit názvy prostředků, musíte také změnit odpovídající hodnotu v klientské aplikaci v souboru *MQ_INSTALLATION_PATH*\tools\wcf\samples\WAS\default\client\app.config a na aplikaci služby v produktu *MQ_INSTALLATION_PATH*\tools\wcf\samples\WAS\HelloWorldsEJBEAR.ear , kde *MQ_INSTALLATION_PATH* je instalační adresář produktu WebSphere MQ. Další informace o formátování identifikátoru URI koncového bodu rozhraní JMS naleznete v tématu Syntaxe identifikátoru URI a parametry pro implementaci webové služby .

Služba a klient jsou založeny na službě a klientovi nastíněné v článku IBM Developer *Sestavení webové služby JMS pomocí protokolu SOAP prostřednictvím rozhraní JMS a produktu WebSphere Studio*. Chcete-li se dozvědět více o vývoji webových služeb SOAP prostřednictvím rozhraní JMS, které jsou kompatibilní s vlastním kanálem produktu WebSphere MQ WCF, lze tento článek nalézt na adrese: https://www.ibm.com/developerworks/websphere/library/techarticles/0402_du/0402_du.html.

Postup

Spusťte klienta jednou: Spusťte soubor *MQ_INSTALLATION_PATH*\tools\wcf\samples\WAS\default\client\bin\Release\TestClient.exe , kde *MQ_INSTALLATION_PATH* je instalační adresář produktu WebSphere MQ.

Klientská aplikace spouští současně obě metody služby a odesílá dvě zprávy do ukázkové fronty.

Výsledky

Aplikace služeb získává zprávy z ukázkové fronty a poskytuje odezvu na metodu HelloWorld(...) , kterou jsou výstupem aplikace klienta na konzolu.

Určování problémů s vlastním kanálem WCF pro produkt WebSphere MQ

Pomocí trasování produktu WebSphere MQ můžete shromažďovat podrobné informace o tom, které různé části kódu produktu WebSphere MQ se budou provádět. Při použití služby WCF (Windows Communication Foundation) je generován samostatný výstup trasování pro trasování vlastního kanálu WCF, který je integrován s trasováním infrastruktury Microsoft WCF.

Úplné zpřístupňování trasování pro vlastní kanál WCF produkuje dva výstupní soubory:

1. Vlastní trasování kanálu WCF bylo integrováno s trasováním infrastruktury Microsoft WCF.
2. Vlastní trasování kanálu služby WCF bylo integrováno s rozhraním XMS .NET.

Pokud máte dva výstupy trasování, problémy lze sledovat na každém rozhraní pomocí příslušných nástrojů, například:

- Určování problémů se WCF pomocí vhodných nástrojů Microsoft .
- Problémy klienta WebSphere MQ MQI s použitím trasovacího formátu XMS .

Chcete-li zjednodušit povolení trasování, je trasovací sada .NET 3 TraceSource a XMS .NET řízena pomocí jednoho rozhraní, jak je popsáno v: “Konfigurace trasování WCF a názvy souborů trasování” na stránce 602.

Hierarchie výjimek vlastního kanálu WCF

Typy výjimek vyvolané vlastním kanálem jsou konzistentní s WCF a obvykle se jedná o výjimku TimeoutException nebo CommunicationException (nebo podtřídy CommunicationException).

Další podrobnosti o chybovém stavu, jsou-li k dispozici, jsou poskytovány pomocí propojených nebo vnitřních výjimek. Následující výjimky jsou typickými příklady a každá vrstva v architektuře kanálu přispívá k další propojené výjimce, například výjimka CommunicationsException má propojenou výjimku XMSException, která má propojenou výjimku MQException:

1. System.serviceModel.CommunicationsExceptions
2. IBM.XMS.XMSException
3. IBM.WMQ.MQException

Informace o klíči jsou zachyceny a poskytnuty v kolekci dat nejvyšší CommunicationException v hierarchii. Zachycení a zajištění dat zabraňuje v tom, aby aplikace propojují s každou vrstvou v architektuře kanálu, aby bylo možné vyslýchat propojené výjimky a všechny další informace, které mohou obsahovat. Jsou definovány následující názvy kláves:

- IBM.XMS.WCF.ErrorCode: Kód chybové zprávy o aktuální výjimce vlastního kanálu.
- IBM.XMS.ErrorCode: Chybová zpráva první výjimky XMS v zásobníku.
- IBM.WMQ.ReasonCode: Základní kód příčiny produktu WebSphere MQ .
- IBM.WMQ.CompletionCode: Základní kód dokončení WebSphere MQ .

Konfigurace trasování WCF a názvy souborů trasování

Je-li trasování plně povoleno, vytvoří dva výstupní soubory, jeden pro diagnostiku problémů WCF a jeden podrobný soubor pro vnitřní diagnostický materiál pro trasování. Chcete-li zjednodušit povolení trasování, budou sady trasování .NET 3 TraceSource i XMS .NET používat jediné rozhraní.

Pro vlastní kanál WCF jsou k dispozici dvě různé metody trasování, tyto dvě metody trasování jsou aktivovány nezávisle nebo společně. Každá metoda vytvoří svůj vlastní trasovací soubor, takže když jsou obě metody trasování aktivovány, vygenerují se dva výstupní soubory trasování.

Chcete-li udržet konfiguraci a povolení co nejjednodušší, použije se stejné rozhraní k řízení obou metod trasování. Soubor `app.config` musí být upraven tak, aby zahrnoval příslušnou konfiguraci trasování, jak je popsáno v následující sekci. Uživatelé pak mohou přidávat své vlastní ekvivalentní sekce ke sloučení výstupu s trasováním ze své vlastní aplikace.

Vlastní trasování kanálu WCF není ve výchozím nastavení povoleno. Nejprve je třeba vytvořit modul listener pro trasování a poté nastavit požadovanou úroveň trasování pro vybraný zdroj trasování v souboru `app.config` .

Konfigurace vlastního kanálu WCF s trasováním infrastruktury WCF

Přidejte následující sekci kódu do sekce `<system.diagnostics><sources>` v souboru `app.config` :

```
<source name="IBM.XMS.WCF" switchValue="Verbose,ActivityTracing">
    <listeners>
        <remove name="Default"/>
        <add name="NewListener"/>
    </listeners>
</source>
```

Předchozí část kódu způsobí, že trasování kanálu bude používat rozhraní .NET 3 TraceSource. Všechna vyvolání konfiguračních souborů přidružených ke spustitelným souborům jsou řízena touto částí kódu.

Konfigurace vlastního kanálu WCF s trasováním XMS .NET

Konfigurace trasování rozhraní XMS .NET vyžaduje, abyste přidali sekci kódu do sekce `<system.diagnostics><sources>` v souboru `app.config` . Část kódu se však přidá do rozšiřitelného prvku `<source>` zobrazeného v sekci Konfigurace vlastního kanálu WCF s trasováním infrastruktury WCF . Takže ačkoli se musí nacházet trasovací kód infrastruktury WCF pro trasování platformy XMS .NET, trasování infrastruktury WCF lze zakázat, pokud není vyžadováno, jak je popsáno v části Povolení trasování WCF .

```
<source name="IBM.XMS.WCF" switchValue="Verbose, ActivityTracing"
 xmsTraceSpecification="*=all=enabled" xmsTraceFilePath="path"
 xmsTraceFileSize="2000000" xmsTraceFileNumber="4" xmsTraceFormat="advanced">
    <listeners>
        <remove name="Default"/>
        <add name="NewListener"/>
```

```
    </listeners>
</source>
```

Konfigurační proměnné trasování WCF

Tabulka 78. Konfigurační proměnné trasování WCF

Proměnná	**Popis**
název	Zadejte název jako: `IBM.XMS.WCF`
switchValue	Úroveň trasování řídí switchValue . Je-li parametr switchValue nastaven na hodnotu `Off`, infrastruktura WCF TraceSource se nevygeneruje. Jakákoli jiná hodnota, jako např. `Verbose`, vygeneruje TraceSource. Podrobné informace o úrovni trasování od společnosti Microsoftnaleznete v dokumentaci k produktu WCF nebo na webové stránce Microsoft WCF Tracing na webové stránce: https://msdn.microsoft.com/en-us/library/ms733025(vs.85).aspx

xmsTraceSpecifikace =*ComponentName=type=state*	*ComponentName* je název třídy, kterou chcete trasovat. V tomto názvu můžete použít zástupný znak *. Příklad:
	`*=all=enabled`
	určuje, že chcete trasovat všechny třídy, a
	`IBM.XMS.impl.*=all=enabled`
	uvádí, že požadujete pouze trasování rozhraní API.
	Typ *typ* může být libovolný z následujících typů trasování:
	• vše
	• ladění
	• událost
	• EntryExit
	Stav *stav* může být povolen nebo zakázán.

xmsTraceFilePath= "*název_souboru*"	Pokud neuvedete xmsTraceFilePath, nebo pokud je xmsTraceFilePath přítomen, ale obsahuje prázdný řetězec, pak trasovací soubor se umístí do aktuálního adresáře. Chcete-li uložit trasovací soubor do uvedeného adresáře, zadejte v souboru xmsTraceFilePathnázev adresáře, například: `xmsTraceFilePath="c:\somepath"`
xmsTraceFileSize= "*velikost*"	Maximální povolená velikost trasovacího souboru. Když soubor dosáhne této velikosti, je archivován a přejmenován. Výchozí maximum je 20 KB, což je uvedeno jako: `xmsTraceFileSize="20000000".`

Tabulka 78. Konfigurační proměnné trasování WCF (pokračování)	
Proměnná	**Popis**
xmsTraceFileNumber= "*číslo*"	Počet trasovacích souborů, které mají být uchovány. Předvolba je 4 (jeden aktivní soubor a tři archivní soubory). Minimální povolený počet je dva.

xmsTraceFormat="*formát*"	Existují dvě úrovně formátu xmsTrace: `basic` a `advanced`. Výchozí formát trasování je základní, pokud nezadáte formát xmsTrace, nebo pokud je přítomen formát xmsTrace, ale obsahuje prázdný řetězec. Trasovací soubory jsou vytvářeny v tomto formátu, pokud zadáte:
	`xmsTraceFormat="basic"`
	Pokud vyžadujete trasování, které je kompatibilní s nástroji pro analýzu trasování, musíte uvést:
	`traceFormat="advanced"`

Povolení trasování WCF

Pro povolení a zakázání dvou různých metod trasování jsou k dispozici čtyři kombinace. Tyto čtyři kombinace vyžadují úpravu hodnot oddílů kódu popsaných v předchozích sekcích.

Je zde také proměnná prostředí, kterou lze nastavit; další informace viz “Povolení trasování WCF pomocí proměnné prostředí WCF_TRACE_ON” na stránce 605.

Tato tabulka a zobrazené hodnoty závisí na níže uvedených částech kódu, které již byly přidány do souboru `app.config` .

Tabulka 79. Kombinace povolení trasování WCF.		
Typ trasování	**Hodnota změněna**	**Příklad**
Trasování XMS je povoleno. WCF TraceSource je povoleno	Hodnota `switchValue` není nastavena na hodnotu `Vypnuto` .	`<source name="IBM.XMS.WCF" switchValue="Verbose, ActivityTracing" xmsTraceSpecification="*=all=enabled" xmsTraceFilePath="path" xmsTraceFileSize="2000000" xmsTraceFileNumber="4" xmsTraceFormat="advanced"> <listeners> <remove name="Default"/> <add name="NewListener"/> </listeners> </source>`
Trasování XMS je povoleno. WCF TraceSource je zakázaný	Parametr `switchValue` je nastaven na hodnotu `Vypnuto` a byl zadán `xmsTraceSpecification` .	`<source name="IBM.XMS.WCF" switchValue="Off, ActivityTracing" xmsTraceSpecification="*=all=enabled" xmsTraceFilePath="path" xmsTraceFileSize="2000000" xmsTraceFileNumber="4" xmsTraceFormat="advanced"> <listeners> <remove name="Default"/> <add name="NewListener"/> </listeners> </source>`

Tabulka 79. Kombinace povolení trasování WCF. (pokračování)

Typ trasování	Hodnota změněna	Příklad
Trasování XMS je zakázáno. WCF TraceSource je povoleno	Tento výsledek lze dosáhnout dvěma způsoby: • Proměnná `switchValue` není nastavena na hodnotu `Off` a položka`xmsTraceSpecificati` on nebyla přidána. • Proměnná `switchValue` není nastavena na hodnotu `Off` a hodnota `xmsTraceSpecification` byla nastavena na hodnotu `disabled` .	`<source name="IBM.XMS.WCF" switchValue="Verbose, ActivityTracing" xmsTraceSpecification="*=all=disabled" xmsTraceFilePath="path" xmsTraceFileSize="2000000" xmsTraceFileNumber="4" xmsTraceFormat="advanced"> <listeners> <remove name="Default"/> <add name="NewListener"/> </listeners> </source>`
Trasování XMS je zakázáno. WCF TraceSource je zakázaný	Existují tři způsoby, jak dosáhnout tohoto výsledku: • V souboru `app.config` není žádný prvek `<source>` . • Proměnná `switchValue` je nastavena na hodnotu `Off` a položka`xmsTraceSpecificati` on nebyla přidána. • Proměnná `switchValue` je nastavena na hodnotu `Off` a parametr `xmsTraceSpecification` byl nastaven na hodnotu `disabled` .	`<source name="IBM.XMS.WCF" switchValue="Off, ActivityTracing" xmsTraceSpecification="*=all=disabled" xmsTraceFilePath="path" xmsTraceFileSize="2000000" xmsTraceFileNumber="4" xmsTraceFormat="advanced"> <listeners> <remove name="Default"/> <add name="NewListener"/> </listeners> </source>`

Povolení trasování WCF pomocí proměnné prostředí WCF_TRACE_ON

Stejně jako předchozí metody popsané v povolení trasování WCF lze trasování rozhraní XMS .NET povolit také s použitím proměnné prostředí WCF_TRACE_ON.

Nastavení proměnné prostředí WCF_TRACE_ON na jinou hodnotu než null je ekvivalentem nastavení hodnoty `xmstraceSpecification` na `*=all=enabled`, například: `"set WCF_TRACE_ON=true"`

Pokud je však parametr `xmstraceSpecification` explicitně nastaven v souboru `app.config` , bude přepsána proměnná prostředí WCF_TRACE_ON.

Výstupní soubory trasování WCF a názvy souborů

Trasovací soubory XMS jsou tradičně pojmenovány s použitím základního názvu a formátu ID procesu: `xms_trace_`*`pid`*`.log`, kde *pid* je ID procesu.

Protože mohou být trasovací soubory XMS stále produkovány paralelně s vlastními soubory trasování kanálu služby WCF, má vlastní trasování kanálu WCF integrované s trasovými soubory trasování XMS .NET následující formát, aby se předešlo nejasnostem: `wcfxms_trace_`*`pid`*`.log`, kde *pid* je ID procesu.

Výstupní soubor trasování je při výchozím nastavení vytvořen v aktuálním pracovním adresáři, ale toto místo určení lze v případě potřeby předefinovat.

WCF XMS First Failure Support Technology (FFST)

Můžete shromažďovat podrobné informace o tom, co různé části kódu produktu WebSphere MQ provádí pomocí trasování produktu WebSphere MQ . XMS FFST má své vlastní konfigurační a výstupní soubory pro vlastní kanál WCF.

Trasovací soubory produktu XMS FFST jsou tradičně pojmenovány s použitím základního názvu a formátu ID procesu: `xmsffdc`*`pid_date`*`.txt`, kde *PID* je ID procesu a *datum* je čas a datum.

Vzhledem k tomu, že soubory trasování produktu XMS FFST mohou být stále vytvářeny paralelně s vlastními soubory XMS FFST kanálů WCF, mají výstupní soubory vlastního kanálu produktu WCF XMS FFST následující formát, aby nedošlo k záměně: `wcfffdc`*`pid_date`*`.txt`, kde *PID* je ID procesu a *datum* je čas a datum.

Tento trasovací výstupní soubor je při výchozím nastavení vytvořen v aktuálním pracovním adresáři, ale toto místo určení lze v případě potřeby předefinovat.

Vlastní kanál WCF se záhlavím trasování .NET XMS je podobný jako v následujícím příkladu:

```
************* Start Display XMS WCF Environment *************
  Product Name  :- value
  WCF Version   :- value
  Level    :- value
*************  End Display XMS WCF Environment  *************
```

Trasovací soubory FFST jsou formátovány standardním způsobem, bez formátování, které je specifické pro vlastní kanál.

Informace o verzi WCF

Informace o verzi WCF pomáhá při určování problémů a je zahrnuta v metadatech sestavení vlastního kanálu.

Vlastní kanál produktu WebSphere MQ pro metadata verze WCF lze načíst jedním ze tří způsobů:

- Pomocí obslužného programu WebSphere MQ dspmqver. Informace o tom, jak používat příkaz dspmqver naleznete v následujícím tématu: dspmqver
- Použití dialogového okna vlastností Průzkumníka Windows : V Průzkumníku Windows klepněte pravým tlačítkem myši na **IBM.XMS.WCF.dll** > **Vlastnosti** > **Verze**.
- Z informací záhlaví libovolného z kanálů FFST nebo trasovacích souborů. Další informace o informacích záhlaví FFST naleznete v tématu: “WCF XMS First Failure Support Technology (FFST)” na stránce 605

Rady a tipy WCF

Následující rady a tipy nejsou v žádném významném pořadí a mohou být přidány do té doby, kdy jsou uvolněny nové verze dokumentace. Jsou to témata, která vám mohou ušetřit čas, pokud jsou důležitá pro práci, kterou právě děláte.

Externalizace výjimek z hostitele služby WCF

Pro služby hostované pomocí hostitele služby WCF nejsou standardně externalizovány všechny neošetřené výjimky vyvolané službou, interními objekty WCF nebo zásobníkem kanálu. Chcete-li být informováni o těchto výjimkách, musí být registrována obslužná rutina chyb.

Následující kód poskytuje příklad definování chování služby obslužné rutiny chyb, které může být použito jako atribut služby:

```
using System.ServiceModel.Dispatcher;
using System.Collections.ObjectModel;
....
    public class ErrorHandlerBehaviorAttribute : Attribute, IServiceBehavior, IErrorHandler
    {
        //
        // IServiceBehavior Interface
        //
        public void AddBindingParameters(ServiceDescription serviceDescription,
           ServiceHostBase serviceHostBase, Collection<ServiceEndpoint> endpoints,
           BindingParameterCollection bindingParameters)
        {
        }
        public void ApplyDispatchBehavior(ServiceDescription serviceDescription,
            ServiceHostBase serviceHostBase)
```

```
        {
            foreach (ChannelDispatcher channelDispatcher in serviceHostBase.ChannelDispatchers)
{
                channelDispatcher.ErrorHandlers.Add(this);
            }
        }
        public void Validate(ServiceDescription serviceDescription, ServiceHostBase
serviceHostBase)
        {
        }

        //
        // IErrorHandler Interface
        //
        public bool HandleError(Exception e)
        {
            // Process the exception in the required way, in this case just outputting to the
console
            Console.Out.WriteLine(e);

            // Always return false to allow any other error handlers to run
            return false;
        }
        public void ProvideFault(Exception error, MessageVersion version, ref Message fault)
        {
        }
    }
```

Ošetřování různých názvů prvků odezvy SOAP

WCF očekává, že název navrácené hodnoty bude standardně ve specifickém formátu, ale služba nemusí vrátit prvek s jeho jménem v očekávaném formátu.

WCF má konvenci očekávající, že vrácená hodnota má být pojmenována v následujícím formátu: *methodName*`Result` , kde *methodName* je název operace služby. Například u služby s názvem `getQuote`očekává požadavek WCF odezvu, která má být volána: `getQuoteResult`.

Služba však může vrátit prvek s názvem, který není v souladu s tímto formátem.

Při spuštění nástroje scvutil pro generování klienta proxy, pokud WSDL uvádí jiný název, přidá rozhraní serveru proxy parametry pro instrukci WCF se jménem, na které se má hledat. Příklad:

```
[System.ServiceModel.OperationContractAttribute(Action = "", ReplyAction = "*")]
[System.ServiceModel.XmlSerializerFormatAttribute(Style =
System.ServiceModel.OperationFormatStyle.Rpc,
                                                  Use =
System.ServiceModel.OperationFormatUse.Encoded)]
[return: System.ServiceModel.MessageParameterAttribute(Name = "getQuoteReturn")]
float getQuote(string in0);
```

Pokud vytvoříte vlastní rozhraní (například přidáním metody požadavek-odezva na existující rozhraní serveru proxy), musíte zajistit, abyste přidali stejné parametry do rozhraní, pokud služba vrátí jiný název. Pokud tak neuděláte, pak nejčastějším problémem je, že volání metody služby vždy vrátí hodnotu null; je-li objekt vrácen, pak metoda vrací hodnotu null, ale pokud je vrácena numerická hodnota, jako je celé číslo, pak metoda vrací 0.

Použití C++

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

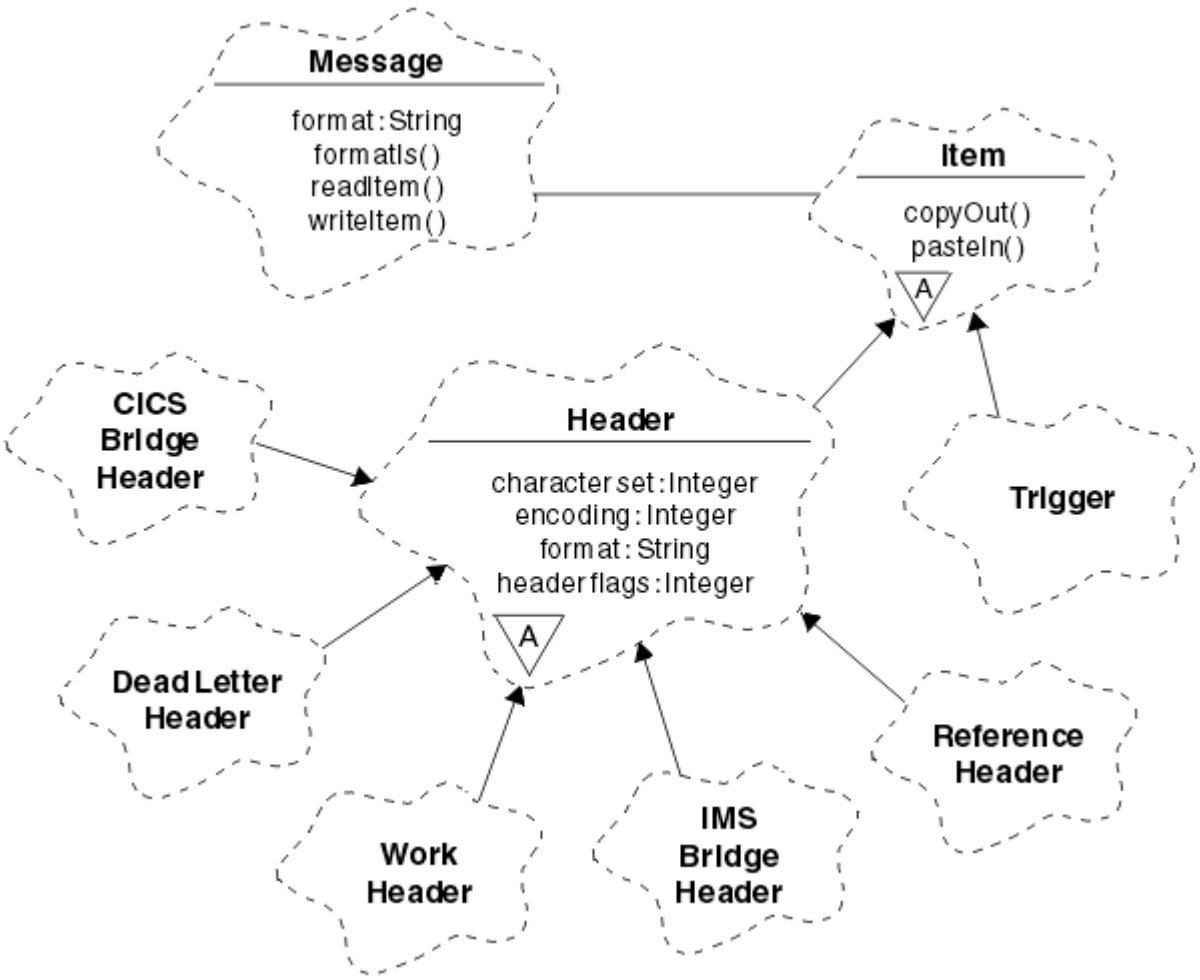
Od produktu WebSphere MQ verze 7.0nebudou rozšíření pro programovací rozhraní produktu WebSphere MQ použita pro třídy C + +.

Produkt WebSphere MQ C++ poskytuje následující funkce:

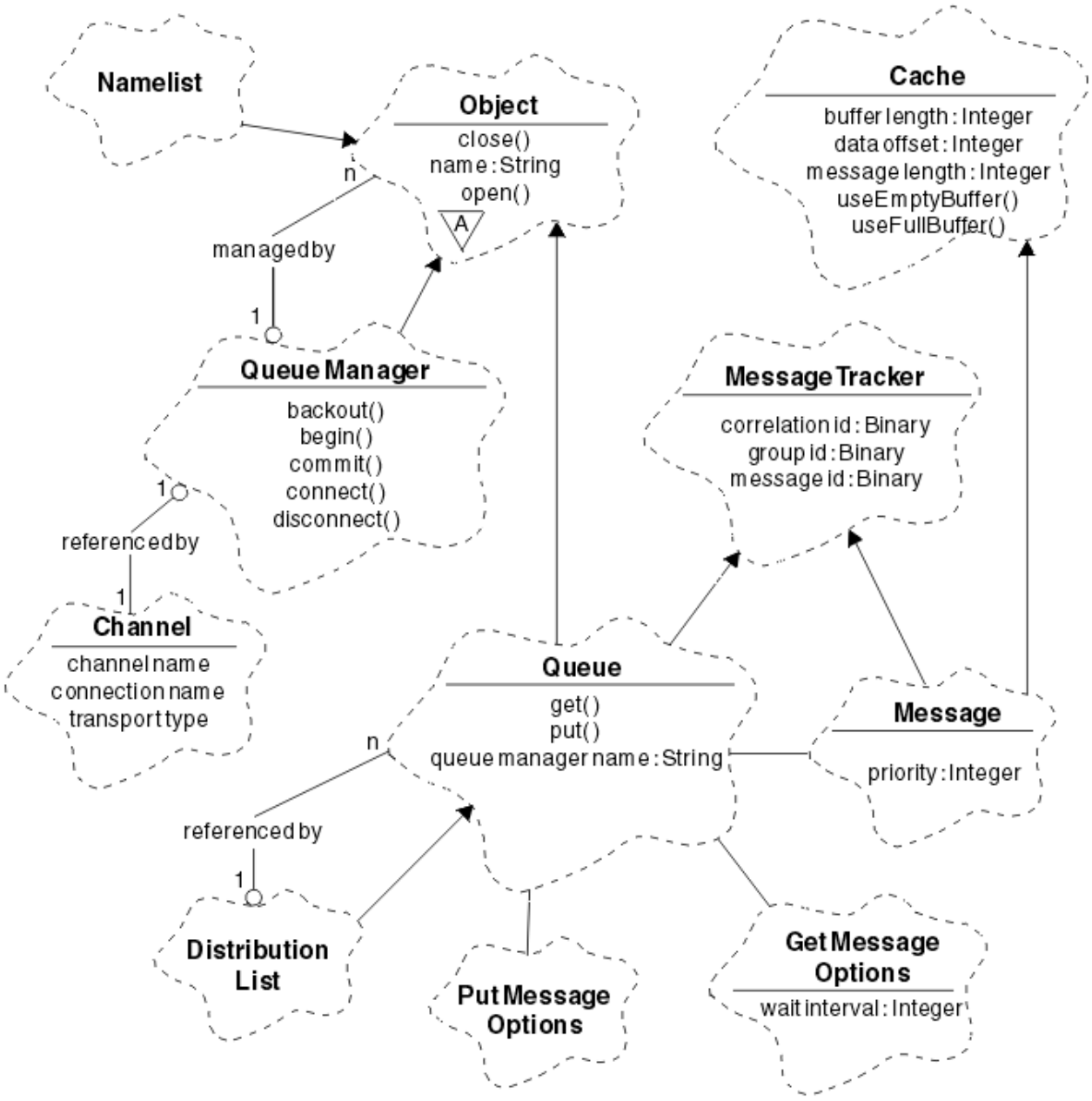
- Automatická inicializace datových struktur produktu WebSphere MQ .
- Otevření fronty a otevření fronty správce front s časem ukončení.
- Odpojení implicitního ukončení fronty a správce front.

- Přenos a příjem záhlaví nedoručených zpráv.
- Přenos a příjem záhlaví mostu produktu IMS .
- Referenční přenos a potvrzení záhlaví zprávy.
- Potvrzení přijetí zprávy.
- Přenos a příjem záhlaví mostu CICS .
- Přenos a příjem záhlaví práce.
- Definice kanálu klienta.

Následující diagramy třídy Booch ukazují, že všechny třídy jsou v podstatě rovnoběžné s entitami WebSphere MQ v procedurálním rozhraní MQI (například pomocí C), které mají buď popisovače, nebo datové struktury. Všechny třídy dědí z třídy ImqError třídy (viz ImqError C++ třída), což umožňuje přiřadit chybový stav ke každému objektu.



Obrázek 120. Třídy C++ produktu WebSphere MQ (zpracování položek)



Obrázek 121. Třídy WebSphere MQ C++ (správa front)

Chcete-li správně interpretovat snímky třídy Booch, uvědomte si následující konvence:

- Metody a zajímavé atributy jsou zobrazeny pod názvem *class* .
- Malý trojúhelník v rámci cloudu označuje *abstraktní třídu* .
- *Dědičnost* je označena šipkou pro nadřízenou třídu.
- Nezdobená čára mezi mraky označuje *kooperativní vztah* mezi třídami.
- Čára zdobená číslem označuje *referenční vztah* mezi dvěma třídami. Číslo označuje počet objektů, které se mohou podílet na určitém vztahu najednou.

Následující třídy a datové typy jsou použity v signatur metody C + + u tříd správy front (viz Obrázek 121 na stránce 609) a tříd zpracování položek (viz Obrázek 120 na stránce 608):

- Třída ImqBinary (viz ImqBinary C++ class), která zapouzdřuje bajtová pole, jako například MQBYTE24.
- Datový typ ImqBoolean , který je definován jako **typedef unsigned char ImqBoolean**.
- Třída ImqString (viz ImqString C++ class), která zapouzdřuje znaková pole jako MQCHAR64.

Entity s datovými strukturami jsou v rámci odpovídajících tříd objektů podsunuté. Jednotlivá pole datové struktury (viz Křížový odkaz jazyka C++ a MQI) jsou přístupná pomocí metod.

Entity s manipulátory spadají do hierarchie tříd ImqObject (viz ImqObject C++ class) a poskytují zapouzdřená rozhraní do rozhraní MQI. Objekty těchto tříd vykazují inteligentní chování, které může snížit počet vyvolání metody vyžadovaných vzhledem k procedurálnímu rozhraní MQI. Můžete například vytvořit a vyřadit připojení správce front podle potřeby, nebo můžete frontu s odpovídajícími volbami otevřít a zavřít ji.

Třída ImqMessage (viz ImqMessage C++ class) zapouzdřuje datovou strukturu MQMD a slouží také jako zadržovací bod pro uživatelská data a *položky* (viz “Čtení zpráv v C++” na stránce 619) poskytnutím prostředků vyrovnávací paměti uložených v mezipaměti. Pro uživatelská data můžete poskytovat vyrovnávací paměti s pevnou délkou a použít ji mnohokrát. Množství dat přítomných ve vyrovnávací paměti se může lišit od jednoho použití k dalšímu. Alternativně může systém poskytovat a spravovat vyrovnávací paměť flexibilní délky. Významné úvahy se stanou velikostí vyrovnávací paměti (dostupné pro příjem zpráv) a skutečně použitou částkou (buď počet bajtů pro přenos, nebo počet skutečně přijatých bajtů).

Související pojmy

Technický přehled

“Ukázkové programy C++” na stránce 610
K dispozici jsou čtyři vzorové programy, které demonstrují získávání a vkládání zpráv.

“Pokyny k jazyku C++” na stránce 614
Tato kolekce témat podrobně popisuje aspekty použití jazyka C++ a konvence, které musíte zvážit při zápisu aplikačních programů, které používají rozhraní MQI (Message Queue Interface).

“Příprava dat zprávy v jazyce C++” na stránce 618
Data zprávy jsou připravena ve vyrovnávací paměti, kterou může poskytnout systém nebo aplikace. Pro obě metody existují výhody. Příklady použití vyrovnávací paměti jsou uvedeny.

“Rozhodování o tom, jaký programovací jazyk použít” na stránce 75
Tyto informace použijte k vyhledání informací o programovacích jazycích a rámcích podporovaných produktem IBM WebSphere MQa o některých aspektech jejich použití.

“Vývoj aplikací” na stránce 7
Produkt IBM WebSphere MQ nabízí několik způsobů, jak vyvíjet aplikace k odesílání a přijímání zpráv, které potřebujete k podpoře vašich obchodních procesů. Také můžete vyvíjet aplikace pro správu správců front a souvisejících prostředků.

Související odkazy

“Sestavování programů jazyka C++ produktu WebSphere MQ” na stránce 624
Adresa URL podporovaných kompilátorů je uvedena spolu s příkazy, které se používají ke kompilaci, propojení a spuštění programů C++ a ukázek na platformách WebSphere MQ .

Křížový odkaz jazyka C++ a MQI

WebSphere MQ Třídy C++

Ukázkové programy C++

K dispozici jsou čtyři vzorové programy, které demonstrují získávání a vkládání zpráv.

Ukázkové programy jsou:

- HELLO WORLD (imqwrld.cpp)
- SPUT (imqsput.cpp)
- SGET (imqsget.cpp)
- DPUT (imqdput.cpp)

Ukázkové programy jsou umístěny v adresářích uvedených v Tabulka 80 na stránce 611.

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

Tabulka 80. Umístění ukázkových programů

Prostředí	Adresář obsahující zdroj	Adresář obsahující sestavení programy
AIX	*MQ_INSTALLATION_PATH*/samp	*MQ_INSTALLATION_PATH*/samp/bin/ia
HP-UX	*MQ_INSTALLATION_PATH*/samp	*MQ_INSTALLATION_PATH*/ samp/bin/ah (viz poznámka “2” na stránce 611)
Solaris	*MQ_INSTALLATION_PATH*/samp	*MQ_INSTALLATION_PATH*/samp/bin/as
Linux	*MQ_INSTALLATION_PATH*/samp	*MQ_INSTALLATION_PATH*/samp/bin/
Windows	*MQ_INSTALLATION_PATH*\tools\cplus\ samples	*MQ_INSTALLATION_PATH*\tools\cplus\ ukázky \bin\vn (viz poznámka “3” na stránce 611)

Notes:

1. Programy sestavené pomocí kompilátoru ILE C++ pro produkt IBM i jsou v knihovně QMQM. Soubory začlenění jsou v /QIBM/ProdData/mqm/inc.
2. Programy sestavené pomocí kompilátoru HP ANSI C + + se nacházejí v adresáři *MQ_INSTALLATION_PATH*/samp/bin/ah. Další informace uvádí téma “Sestavování programů C++ v systému HP-UX” na stránce 625.
3. Programy sestavené pomocí produktu Microsoft Visual Studio se nacházejí v adresáři *MQ_INSTALLATION_PATH*\tools\cplus\samples\bin\vn. Další informace o těchto kompilátorech naleznete v tématu “Sestavování programů C++ v systému Windows” na stránce 631.

Ukázkový program HELLO WORLD (imqwrld.cpp)

Tento ukázkový program C++ ukazuje, jak vložit a získat regulární datagram (strukturu C) pomocí třídy ImqMessage .

Tento program ukazuje, jak vložit a získat regulární datagram (strukturu C) pomocí třídy ImqMessage . Tato ukázka používá málo vyvolání metody, přičemž využívá implicitní vyvolání metod, jako jsou **otevření**, **zavření**a **odpojení**.

Na všech platformách s výjimkou produktu z/OS

Používáte-li připojení serveru k produktu WebSphere MQ, postupujte podle jedné z následujících procedur:

- Chcete-li použít existující výchozí frontu, SYSTEM.DEFAULT.LOCAL.QUEUE, spusťte program **imqwrlds** bez předání jakýchkoli parametrů
- Chcete-li použít dočasnou dynamicky přiřazenou frontu, spusťte příkaz **imqwrlds** s předáním názvu výchozí modelové fronty SYSTEM.DEFAULT.MODEL.QUEUE.

Používáte-li připojení klienta k produktu WebSphere MQ, postupujte podle jedné z následujících procedur:

- Nastavte proměnnou prostředí MQSERVER (viz MQSERVER , kde získáte další informace) a spusťte příkaz **imqwrldc**nebo
- Příkaz **imqwrldc** se spustí jako parametry **queue-name**, **queue-manager-name**a **channel-definition**, kde typická hodnota **channel-definition** může být SYSTEM.DEF.SVRCONN/TCP/ *název_hostitele*(1414)

Ukázkový kód

```
extern "C" {
#include <stdio.h>
}

#include <imqi.hpp> // WebSphere MQ C++

#define EXISTING_QUEUE "SYSTEM.DEFAULT.LOCAL.QUEUE"

#define BUFFER_SIZE 12

static char gpszHello[ BUFFER_SIZE ] = "Hello world" ;
int main ( int argc, char * * argv ) {
  ImqQueueManager manager ;
  int iReturnCode = 0 ;

  // Connect to the queue manager.
  if ( argc > 2 ) {
    manager.setName( argv[ 2 ] );
  }
  if ( manager.connect( ) ) {
    ImqQueue * pqueue = new ImqQueue ;
    ImqMessage * pmsg = new ImqMessage ;

    // Identify the queue which will hold the message.
    pqueue -> setConnectionReference( manager );
    if ( argc > 1 ) {
      pqueue -> setName( argv[ 1 ] );


      // The named queue can be a model queue, which will result in
      // the creation of a temporary dynamic queue, which will be
      // destroyed as soon as it is closed. Therefore we must ensure
      // that such a queue is not automatically closed and reopened.
      // We do this by setting open options which will avoid the need
      // for closure and reopening.
      pqueue -> setOpenOptions( MQOO_OUTPUT | MQOO_INPUT_SHARED |
                                MQOO_INQUIRE );
    } else {
      pqueue -> setName( EXISTING_QUEUE );

      // The existing queue is not a model queue, and will not be
      // destroyed by automatic closure and reopening. Therefore we
      // will let the open options be selected on an as-needed basis.
      // The queue will be opened implicitly with an output option
      // during the "put", and then implicitly closed and reopened
      // with the addition of an input option during the "get".
    }

    // Prepare a message containing the text "Hello world".
    pmsg -> useFullBuffer( gpszHello , BUFFER_SIZE );
    pmsg -> setFormat( MQFMT_STRING );

    // Place the message on the queue, using default put message
    // Options.
    // The queue will be automatically opened with an output option.
    if ( pqueue -> put( * pmsg ) ) {
      ImqString strQueue( pqueue -> name( ) );

      // Discover the name of the queue manager.
      ImqString strQueueManagerName( manager.name( ) );
      printf( "The queue manager name is %s.\n",
              (char *)strQueueManagerName );

      // Show the name of the queue.
      printf( "Message sent to %s.\n", (char *)strQueue );

      // Retrieve the data message just sent ("Hello world" expected)
      // from the queue, using default get message options. The queue
      // is automatically closed and reopened with an input option
      // if it is not already open with an input option. We get the
      // message just sent, rather than any other message on the
      // queue, because the "put" will have set the ID of the message
      // so, as we are using the same message object, the message ID
      // acts as in the message object, a filter which says that we
      // are interested in a message only if it has this
      // particular ID.

      if ( pqueue -> get( * pmsg ) ) {
```

```
        int iDataLength = pmsg -> dataLength( );

        // Show the text of the received message.
        printf( "Message of length %d received, ", iDataLength );

        if ( pmsg -> formatIs( MQFMT_STRING ) ) {
          char * pszText = pmsg -> bufferPointer( );

          // If the last character of data is a null, then we can
          // assume that the data can be interpreted as a text
          // string.
          if ( ! pszText[ iDataLength - 1 ] ) {
            printf( "text is \"%s\".\n", pszText );
          } else {
            printf( "no text.\n" );
          }

        } else {
          printf( "non-text message.\n" );
        }
      } else {
        printf( "ImqQueue::get failed with reason code %ld\n",
                pqueue -> reasonCode( ) );
        iReturnCode = (int)pqueue -> reasonCode( );
      }

    } else {
      printf( "ImqQueue::open/put failed with reason code %ld\n",
              pqueue -> reasonCode( ) );
      iReturnCode = (int)pqueue -> reasonCode( );
    }

    // Deletion of the queue will ensure that it is closed.
    // If the queue is dynamic then it will also be destroyed.
    delete pqueue ;
    delete pmsg ;

  } else {
    printf( "ImqQueueManager::connect failed with reason code %ld\n"
            manager.reasonCode( ) );
    iReturnCode = (int)manager.reasonCode( );
  }

  // Destruction of the queue manager ensures that it is
  // disconnected.  If the queue object were still available
  // and open (which it is not), the queue would be closed
  // prior to disconnection.

  return iReturnCode ;
}
```

Vzorové programy SPUT (imqsput.cpp) a SGET (imqsget.cpp)

Tyto programy C++ umisťují zprávy do pojmenované fronty a načítají je z fronty.

Tyto ukázky zobrazují použití následujících tříd:

- ImqError (viz ImqError Třída C++)
- ImqMessage (viz ImqMessage C++ class)
- ImqObject (viz ImqObject C++ class)
- ImqQueue (viz ImqQueue C++ class)
- Správce ImqQueueManager (viz ImqQueueManager C++ class)

Postupujte podle příslušných pokynů pro spouštění programů.

Na všech platformách s výjimkou produktu z/OS

1. Spusťte příkaz **imqsputs** *název-fronty*.
2. Zadejte řádky textu na konzole. Tyto řádky jsou umístěny jako zprávy do zadané fronty.
3. Zadejte řádek null, abyste ukončili vstup.

4. Spuštěním příkazu **imqsgets** *název_fronty* načtěte všechny řádky a zobrazte je na konzole.

Ukázkový program DPUT (imqdput.cpp)

Tento ukázkový program C++ umisťuje zprávy do rozdělovníku, který se skládá ze dvou front.

Funkce DPUT zobrazuje použití třídy ImqDistributionList (viz ImqDistributionList C++ class). Tato ukázka není podporována v systému z/OS.

1. Spuštěním příkazu **imqdputs** *queue-name-1 queue-name-2* umístěte zprávy do dvou pojmenovaných front.
2. Spuštěním příkazu **imqsgets** *queue-name-1* a **imqsgets** *queue-name-2* načtěte zprávy z těchto front.

Pokyny k jazyku C++

Tato kolekce témat podrobně popisuje aspekty použití jazyka C++ a konvence, které musíte zvážit při zápisu aplikačních programů, které používají rozhraní MQI (Message Queue Interface).

Soubory záhlaví C++

Soubory záhlaví jsou poskytovány jako součást definice rozhraní MQI, které vám pomohou psát aplikační programy produktu WebSphere MQ v jazyce C + +.

Tyto soubory záhlaví jsou shrnuty v následující tabulce.

Tabulka 81. Soubory záhlaví C/C++

Název souboru	**Obsah**
IMQI.HPP	Třídy rozhraní MQI jazyka C++ (včetně CMQC.H a IMQTYPE.H)
IMQTYPE.H	Definuje datový typ **ImqBoolean** .
CMQC.H	Struktury dat MQI a konstanty souboru typu manifest

Chcete-li zlepšit přenositelnost aplikací, uveďte název souboru záhlaví malými písmeny na direktivě preprocesoru **#include** :

```
#include <imqi.hpp> // C++ classes
```

Metody a atributy jazyka C++

Názvy metod jsou ve smíšeném případě. Na parametry a návratové hodnoty se vztahují různé aspekty. Atributy jsou přístupné pomocí metody set a get, jak je to vhodné.

Parametry metod, které jsou *const* , jsou určeny pouze pro vstup. Parametry s podpisy včetně ukazatele (*) nebo odkazu (&) jsou předávány odkazem. Návratové hodnoty, které nezahrnují ukazatel nebo odkaz, jsou předávány hodnotou; v případě vrácených objektů se jedná o nové entity, které se staly odpovědností volajícího.

Některé signatury metod obsahují položky, které berou výchozí hodnotu, pokud nejsou uvedeny. Takové položky jsou vždy na konci podpisů a jsou označeny rovnítkem (=); hodnota za rovnítkem označuje výchozí hodnotu, která se použije, pokud je položka vynechána.

Všechny názvy metod v těchto třídách jsou malými písmeny, počínaje malými písmeny. Každé slovo, kromě prvního v názvu metody, začíná velkým písmenem. Zkratky se nepoužijí, pokud jejich význam není všeobecně srozumitelná. Použité zkratky zahrnují *id* (pro identitu) a *sync* (pro synchronizaci).

K atributům objektu se přistupuje pomocí metody set a get. Metoda set začíná slovem *set*; metoda get nemá žádnou předponu. Je-li atribut *jen pro čtení*, není nastavena žádná metoda set.

Atributy jsou inicializovány na platné stavy během konstrukce objektu a stav objektu je vždy konzistentní.

Datové typy v C++

Všechny datové typy jsou definovány příkazem C **typedef** .

Typ **ImqBoolean** je definován jako **unsigned char** v IMQTYPE.H a mohou mít hodnoty TRUE a FALSE. Můžete použít objekty třídy **ImqBinary** místo polí **MQBYTE** a objekty třídy **ImqString** na místě **znak ***. Mnoho metod vrací objekty místo znaků **char** nebo **MQBYTE** k usnadnění správy ukládání dat. Všechny návratové hodnoty se stanou odpovědností volajícího a v případě vráceného objektu může být úložiště likvidován pomocí odstranění.

Manipulace s binárními řetězci v jazyce C++

Řetězce binárních dat jsou deklarovány jako objekty třídy **ImqBinary** . Objekty této třídy lze kopírovat, porovnat a nastavit pomocí známých operátorů C. Vzorový kód je poskytnut.

Následující ukázka kódu zobrazuje operace na binárním řetězci:

```
#include <imqi.hpp> // C++ classes

ImqMessage message ;
ImqBinary id, correlationId ;
MQBYTE24 byteId ;

correlationId.set( byteId, sizeof( byteId ) ); // Set.
id = message.id( );                            // Assign.
if ( correlationId == id ) {                   // Compare.
  ...
```

Manipulace se znakovými řetězci v C++

Znaková data jsou často vrácena v objektech třídy **ImqString** , které lze přetypovat na **char *** pomocí operátoru převodu. Třída ImqString obsahuje metody pomáhající při zpracování znakových řetězců.

Jsou-li znaková data přijata nebo vrácena pomocí metod rozhraní MQI C + +, jsou znaková data vždy null-ukončena a mohou mít libovolnou délku. Určité limity však vynucují produkt WebSphere MQ , který může za následek zkrácení informací. Za účelem usnadnění správy ukládání dat se často vrací znaková data do objektů třídy **ImqString** . Tyto objekty lze přetypovat na **char *** pomocí operátoru převodu, který je poskytován, a používá se pro účely *jen pro čtení* v mnoha situacích, kdy je vyžadován znak **char *** .

Poznámka: Výsledek převodu **char *** z objektu třídy **ImqString** může mít hodnotu null.

Ačkoli lze funkce jazyka C použít na **char ***, existují speciální metody třídy **ImqString** , které jsou vhodnější; **operátor length**() je ekvivalentní s **strlen** a **storage**() označuje velikost paměti přidělené pro znaková data.

Initial state of objects in C++

Všechny objekty mají konzistentní počáteční stav, který odráží jejich atributy. Počáteční hodnoty jsou definovány v popisech tříd.

Použití jazyka C z jazyka C++

Používáte-li funkce C z programu C + +, zahrňte příslušná záhlaví.

Následující příklad ukazuje program `string.h` zahrnutý v programu C + +:

```
extern "C" {
#include <string.h>
}
```

Notační konvence C + +

Tento příklad ukazuje, jak vyvolat metody a deklarovat parametry.

Tato ukázka kódu používá metody a parametry**ImqBoolean ImqQueue**::**get**(**ImqMessage &** *msg*).

Deklarujte a použijte parametry následujícím způsobem:

```
ImqQueueManager * pmanager ;    // Queue manager
ImqQueue * pqueue ;             // Message queue
ImqMessage msg ;                // Message
char szBuffer[ 100 ];          // Buffer for message data

pmanager = new ImqQueueManager ;
pqueue = new ImqQueue ;
pqueue -> setName( "myreplyq" );
pqueue -> setConnectionReference( pmanager );

msg.useEmptyBuffer( szBuffer, sizeof( szBuffer ) );

if ( pqueue -> get( msg ) ) {
  long lDataLength = msg.dataLength( );

  ...
}
```

Implicitní operace v C++

Několik operací může nastat implicitně, *právě včas*, aby splnil podmínky splnění předpokladů pro úspěšné provedení metody. Tyto implicitní operace jsou connect, open, reopen, close a disconnect. Můžete řídit připojení a otevřít implicitní chování pomocí atributů třídy.

Připojit

Objekt správce ImqQueueeje připojen automaticky pro libovolnou metodu, která vede k volání MQI (viz Křížový odkaz jazyka C++ a MQI).

Otevřené

Objekt ImqObject je automaticky otevřen pro libovolnou metodu, která vede k volání MQGET, MQINQ, MQPUT nebo MQSET. Použijte metodu **openFor** k uvedení jedné nebo více relevantních hodnot **volby otevření** .

Znovu otevřít

Objekt ImqObject je automaticky znovu otevřen pro libovolnou metodu, jejímž výsledkem je volání MQGET, MQINQ, MQPUT nebo MQSET, kde je objekt již otevřen, ale existující **volby otevření** nejsou dostatečné k tomu, aby bylo volání MQI úspěšné. Objekt je dočasně zavřen pomocí dočasné hodnoty **close options** MQCO_NONE. Použijte metodu **openFor** k přidání relevantního**volba otevření**.

Opětné otevření může způsobit problémy za určitých okolností:

- Dočasná dynamická fronta je zničena, když je uzavřena a nelze ji nikdy znovu otevřít.
- Fronta otevřená pro výlučný vstup (ať už explicitně, nebo při výchozím nastavení) může být přístupná pro jiné osoby v okně příležitost během zavírání a opětovného otevření.
- Poloha kurzoru při procházení se ztratí, když je fronta zavřena. Tato situace nebrání zavření a opětovnému otevření, ale zabrání následnému použití kurzoru, dokud nebude znovu použit MQGMO_BROTE_FIRST.
- Kontext poslední načtené zprávy je ztracen, když je fronta uzavřena.

Pokud se některý z těchto okolností vyskytne nebo může být předvídaný, vyhněte se opětovnému otevření explicitním nastavením vhodných **voleb otevření** před otevřením objektu (ať už explicitně, nebo implicitně).

Nastavení **voleb otevření** explicitně pro složité situace obsluhující fronty vede k lepšímu výkonu a vyhýbá se problémům spojeným s opětným otevřením.

Zavřít
Objekt ImqObject je automaticky zavřen v libovolném okamžiku, kdy stav objektu již není životaschopný, například pokud je odkaz na připojení ImqObject přerušen, nebo je-li objekt ImqObject zničen.

Odpojit
Správce ImqQueueje automaticky odpojen v libovolném okamžiku, kdy připojení již není realizovatelné, například pokud je odkaz na připojení ImqObject přerušený, nebo pokud je objekt ImqQueueManager zničen.

Binární a znakové řetězce v jazyce C++

Třída ImqString zapouzdřuje tradiční datový formát *char ** . Třída ImqBinary zapouzdřuje binární bajtové pole. Některé metody, které nastavují znaková data, by mohly data oříznout.

Metody, které nastavují znak (**char ***), vždy berou kopii dat, ale některé metody mohou tuto kopii zkrátit, protože produkt WebSphere MQuloží určité limity.

Třída ImqString (viz ImqString C++ class) zapouzdřuje tradiční **znak *** a poskytuje podporu pro:

- Porovnání
- Zřetězení
- Kopírování
- převod typu Integer-to-text a text-to-integer
- Extrakce tokenu (slova)
- Překlad velkých písmen

Třída ImqBinary (viz ImqBinary C++ class) zapouzdřuje binární bajtová pole libovolné velikosti. Zejména se používá k uchování následujících atributů:

- **accounting token (účtovací token)** (MQBYTE32)
- **connection tag = příznak připojení** (MQBYTE128)
- **correlation id** (MQBYTE24).
- **token zařízení** (MQBYTE8)
- **group id (ID skupiny)** (MQBYTE24)
- **instance id** (MQBYTE24)
- **message id** (MQBYTE24)
- **token zprávy** (MQBYTE16)
- **ID instance transakce** (MQBYTE16)

Kde tyto atributy patří do objektů v následujících třídách:

- ImqCICSBridgeZáhlaví (viz ImqCICSBridgeHeader C++ class)
- ImqGetMessageOptions (viz ImqGetMessageOptions C++ class).
- Záhlaví ImqIMSBridge(viz ImqIMSBridgeHeader C++ class)
- ImqMessageTracker (viz ImqMessageTracker C++ class)
- Správce ImqQueueManager (viz ImqQueueManager C++ class)
- záhlaví ImqReference(viz ImqReferenceTřída C++ záhlaví)
- Záhlaví ImqWork(viz ImqWorkHeader C++ class)

Třída ImqBinary také poskytuje podporu pro porovnání a kopírování.

Nepodporované funkce v jazyce C++

Třídy a metody produktu WebSphere MQ C++ jsou nezávislé na platformě WebSphere MQ . Mohou tedy nabízet některé funkce, které nejsou na určitých platformách podporovány.

Pokusíte-li se použít funkci na platformě, na které není podporována, funkce je zjištěna produktem WebSphere MQ , ale ne vazbami jazyka C + +. Produkt WebSphere MQ ohlásí chybu vašemu programu, stejně jako jakákoli jiná chyba MQI.

Systém zpráv v C++

Tato kolekce témat obsahuje podrobné informace o přípravě, čtení a zápisu zpráv v jazyce C + +.

Příprava dat zprávy v jazyce C++

Data zprávy jsou připravena ve vyrovnávací paměti, kterou může poskytnout systém nebo aplikace. Pro obě metody existují výhody. Příklady použití vyrovnávací paměti jsou uvedeny.

Při odeslání zprávy se data zprávy nejprve připraví ve vyrovnávací paměti spravované objektem ImqCache (viz ImqCache C++ class). Vyrovnávací paměť je přidružena (podle dědičnosti) k jednotlivým objektům ImqMessage (viz ImqMessage C++ class): může být dodána aplikací (buď pomocí metody **useEmptyBuffer** , nebo **useFullBuffer**) nebo automaticky systémem. Výhoda aplikace dodávající vyrovnávací paměť zpráv znamená, že v mnoha případech není nutné kopírovat žádné datové kopie, protože aplikace může přímo používat připravené datové oblasti. Nevýhodou je, že zadaná vyrovnávací paměť má pevnou délku.

Vyrovnávací paměť lze znovu použít a počet přenesených bajtů lze každou dobu měnit, a to pomocí metody **setMessageLength** před přenosem.

Je-li systém určen automaticky, je počet dostupných bajtů spravován systémem a data lze zkopírovat do vyrovnávací paměti zprávy například pomocí metody ImqCache **write** nebo metody ImqMessage **writeItem** . Velikost vyrovnávací paměti zpráv roste podle potřeby. Jak velikost vyrovnávací paměti narůstá, nedošlo ke ztrátě dříve zapsaných dat. Velká nebo vícedílná zpráva může být zapsána v sekvenčních kouskách.

Následující příklady ukazují zjednodušené odeslání zprávy.

1. Použít připravená data ve vyrovnávací paměti dodané uživatelem

```
char szBuffer[ ] = "Hello world" ;

msg.useFullBuffer( szBuffer, sizeof( szBuffer ) );
msg.setFormat( MQFMT_STRING );
```

2. Použijte připravená data v uživatelem zadané vyrovnávací paměti, kde velikost vyrovnávací paměti překračuje velikost dat.

```
char szBuffer[ 24 ] = "Hello world" ;

msg.useEmptyBuffer( szBuffer, sizeof( szBuffer ) );
msg.setFormat( MQFMT_STRING );
msg.setMessageLength( 12 );
```

3. Kopírovat data do vyrovnávací paměti zadané uživatelem

```
char szBuffer[ 12 ];

msg.useEmptyBuffer( szBuffer, sizeof( szBuffer ) );
msg.setFormat( MQFMT_STRING );
msg.write( 12, "Hello world" );
```

4. Kopírovat data do vyrovnávací paměti dodávané systémem

```
msg.setFormat( MQFMT_STRING );
```

```
msg.write( 12, "Hello world" );
```

5. Kopírovat data do vyrovnávací paměti dodávané systémem s použitím objektů (objekty nastavují formát zprávy a obsah)

```
ImqString strText( "Hello world" );

msg.writeItem( strText );
```

Čtení zpráv v C++

Vyrovnávací paměť může být dodána aplikací nebo systémem. K datům lze přistupovat přímo z vyrovnávací paměti nebo číst postupně. Pro každý typ zprávy existuje třída ekvivalentní. Ukázkový kód je uveden.

Při příjmu dat může aplikace nebo systém dodat vhodnou vyrovnávací paměť zpráv. Stejnou vyrovnávací paměť lze použít pro vícenásobný přenos i pro více příjemku pro konkrétní objekt ImqMessage . Je-li vyrovnávací paměť zpráv dodána automaticky, bude zvětšeno, aby bylo možné přijmout jakoukoli délku dat, která je k dispozici. Avšak vyrovnávací paměť zpráv dodaná aplikací nemusí být dostatečně velká, aby zadržela přijatá data. Potom může dojít k oříznutí nebo selhání, v závislosti na volbách použitých pro přijetí zprávy.

K příchozím datům lze přistupovat přímo z vyrovnávací paměti zpráv, v takovém případě délka dat označuje celkové množství příchozích dat. Jinou možností je, že příchozí data lze číst sekvenčně z vyrovnávací paměti zpráv. V tomto případě se ukazatel na data adresuje dalšímu bajtu příchozích dat a datový ukazatel a délka dat se aktualizují pokaždé, když se čtou data.

Položky jsou části zprávy, vše v uživatelské oblasti vyrovnávací paměti zpráv, které je třeba zpracovávat postupně a samostatně. Kromě běžných uživatelských dat může být položka záhlavím se smrtícím písmenem nebo zprávou spouštěče. Položky jsou vždy přidruženy s formáty zpráv; formáty zpráv jsou ***ne*** vždy přidruženy k položkám.

Pro každou položku, která odpovídá rozpoznatelné struktuře zpráv produktu WebSphere MQ , je objekt třídy objektu. Pro záhlaví a jednu zprávu spouštěče se nachází jedna záhlaví a jedna záhlaví. Pro uživatelská data neexistuje žádná třída objektu. To znamená, že jakmile budou rozpoznatelné formáty vyčerpány, bude zbývající část zpracování ponechána aplikačnímu programu. Třídy pro data uživatelů lze zapisovat prostřednictvím specializace třídy ImqItem .

Následující příklad zobrazuje příjemku zprávy, která bere v úvahu celou řadu potenciálních položek, které mohou předcházet datům uživatele, v imaginární situaci. Uživatelská data, která nejsou položkou, jsou definována jako vše, co nastane za položkami, které lze identifikovat. Automatická vyrovnávací paměť (výchozí) se používá k uložení libovolného množství dat zprávy.

```
ImqQueue queue ;
ImqMessage msg ;

if ( queue.get( msg ) ) {

  /* Process all items of data in the message buffer. */
  do while ( msg.dataLength( ) ) {
    ImqBoolean bFormatKnown = FALSE ;
    /* There remains unprocessed data in the message buffer. */

    /* Determine what kind of item is next. */

    if ( msg.formatIs( MQFMT_DEAD_LETTER_HEADER ) ) {
      ImqDeadLetterHeader header ;
      /* The next item is a dead-letter header.                */
      /* For the next statement to work and return TRUE,       */
      /* the correct class of object pointer must be supplied. */
      bFormatKnown = TRUE ;

      if ( msg.readItem( header ) ) {
```

```
        /* The dead-letter header has been extricated from the */
        /* buffer and transformed into a dead-letter object.   */
        /* The encoding and character set of the dead-letter   */
        /* object itself are MQENC_NATIVE and MQCCSI_Q_MGR.    */
        /* The encoding and character set from the dead-letter */
        /* header have been copied to the message attributes   */
        /* to reflect any remaining data in the buffer.        */

        /* Process the information in the dead-letter object.  */
        /* Note that the encoding and character set have       */
        /* already been processed.                             */
        ...
      }
      /* There might be another item after this, */
      /* or just the user data.                  */
    }
    if ( msg.formatIs( MQFMT_TRIGGER ) ) {
      ImqTrigger trigger ;
      /* The next item is a trigger message.                  */
      /* For the next statement to work and return TRUE,      */
      /* the correct class of object pointer must be supplied. */
      bFormatKnown = TRUE ;
      if ( msg.readItem( trigger ) ) {

        /* The trigger message has been extricated from the */
        /* buffer and transformed into a trigger object.    */
        /* Process the information in the trigger object. */
        ...
      }

      /* There is usually nothing after a trigger message. */
    }

    if ( msg.formatIs( FMT_USERCLASS ) ) {
      UserClass object ;
      /* The next item is an item of a user-defined class.     */
      /* For the next statement to work and return TRUE,       */
      /* the correct class of object pointer must be supplied. */
      bFormatKnown = TRUE ;

      if ( msg.readItem( object ) ) {
        /* The user-defined data has been extricated from the */
        /* buffer and transformed into a user-defined object. */

        /* Process the information in the user-defined object. */
        ...
      }

      /* Continue looking for further items. */
    }
    if ( ! bFormatKnown ) {
      /* There remains data that is not associated with a specific*/
      /* item class.                                               */
      char * pszDataPointer = msg.dataPointer( );       /* Address.*/
      int iDataLength = msg.dataLength( );              /* Length. */

      /* The encoding and character set for the remaining data are */
      /* reflected in the attributes of the message object, even   */
      /* if a dead-letter header was present.                      */
      ...

    }

  }
}
```

V tomto příkladě je FMT_USERCLASS konstantou představující 8znakový název formátu přidružený k objektu třídy `UserClassa` je definován aplikací.

`UserClass` je odvozen z třídy ImqItem (viz ImqItem C++ class) a implementuje metody **copyOut** a **pasteIn** z této třídy.

Následující dva příklady zobrazují kód ze třídy ImqDeadLetterHeader (viz ImqDeadLetterHeader C++ class). První příklad ukazuje vlastní zapouzdřenou zprávu-*writing* kód.

```
// Insert a dead-letter header.
// Return TRUE if successful.
```

```
ImqBoolean ImqDeadLetterHeader :: copyOut ( ImqMessage & msg ) {
  ImqBoolean bSuccess ;
  if ( msg.moreBytes( sizeof( omqdlh ) ) ) {
    ImqCache cacheData( msg ); // Preserve original message content.
    // Note original message attributes in the dead-letter header.
    setEncoding( msg.encoding( ) );
    setCharacterSet( msg.characterSet( ) );
    setFormat( msg.format( ) );

    // Set the message attributes to reflect the dead-letter header.
    msg.setEncoding( MQENC_NATIVE );
    msg.setCharacterSet( MQCCSI_Q_MGR );
    msg.setFormat( MQFMT_DEAD_LETTER_HEADER );
    // Replace the existing data with the dead-letter header.
    msg.clearMessage( );
    if ( msg.write( sizeof( omqdlh ), (char *) & omqdlh ) ) {
      // Append the original message data.
      bSuccess = msg.write( cacheData.messageLength( ),
                            cacheData.bufferPointer( ) );
    } else {
      bSuccess = FALSE ;
    }
  } else {
    bSuccess = FALSE ;
  }
  // Reflect and cache error in this object.
  if ( ! bSuccess ) {
    setReasonCode( msg.reasonCode( ) );
    setCompletionCode( msg.completionCode( ) );
  }

  return bSuccess ;
}
```

Druhý příklad ukazuje vlastní zapouzdřenou zprávu-*reading* kód.

```
// Read a dead-letter header.
// Return TRUE if successful.
ImqBoolean ImqDeadLetterHeader :: pasteIn ( ImqMessage & msg ) {
  ImqBoolean bSuccess = FALSE ;

  // First check that the eye-catcher is correct.
  // This is also our guarantee that the "character set" is correct.
  if ( ImqItem::structureIdIs( MQDLH_STRUC_ID, msg ) ) {
    // Next check that the "encoding" is correct, as the MQDLH
    // contains numeric data.
    if ( msg.encoding( ) == MQENC_NATIVE ) {

      // Finally check that the "format" is correct.
      if ( msg.formatIs( MQFMT_DEAD_LETTER_HEADER ) ) {
        char * pszBuffer = (char *) & omqdlh ;
        // Transfer the MQDLH from the message and move pointer on.
        if ( bSuccess = msg.read( sizeof( omdlh ), pszBuffer ) ) {
          // Update the encoding, character set and format of the
          // message to reflect the remaining data.
          msg.setEncoding( encoding( ) );
          msg.setCharacterSet( characterSet( ) );
          msg.setFormat( format( ) );
        } else {

          // Reflect the cache error in this object.
          setReasonCode( msg.reasonCode( ) );
          setCompletionCode( msg.completionCode( ) );
        }
      } else {
        setReasonCode( MQRC_INCONSISTENT_FORMAT );
        setCompletionCode( MQCC_FAILED );
      }
    } else {
      setReasonCode( MQRC_ENCODING_ERROR );
      setCompletionCode( MQCC_FAILED );
    {
  } else {
    setReasonCode( MQRC_STRUC_ID_ERROR );
    setCompletionCode( MQCC_FAILED );
  }

  return bSuccess ;
```

```
}
```

S automatickým vyrovnávacími paměťmi je vyrovnávací paměť *nestálá*. To znamená, že data vyrovnávací paměti se mohou nacházet v jiném fyzickém umístění po každém vyvolání metody **get** . Proto jsou při každém odkazování na data vyrovnávací paměti použita metoda **bufferPointer** nebo **dataPointer** k přístupu k datům zpráv.

Můžete chtít, aby program vyjmula pevnou oblast pro příjem dat zprávy. V takovém případě vyvolejte metodu **useEmptyBuffer** předtím, než použijete metodu **get** .

Použití pevné a neautomatické oblasti omezuje zprávy na maximální velikost, takže je důležité vzít v úvahu volbu MQGMO_ACCEPT_TRUNCATED_MSG objektu ImqGetMessageOptions . Není-li tato volba zadána (výchozí nastavení), lze očekávat kód příčiny MQRC_TRUNCATED_MSG_FAILED. Je-li zadána tato volba, může být v závislosti na návrhu aplikace očekáván kód příčiny MQRC_TRUNCATED_MSG_ACCEPTED.

Následující příklad ukazuje, jak lze použít pevnou oblast paměti pro příjem zpráv:

```
char * pszBuffer = new char[ 100 ];

msg.useEmptyBuffer( pszBuffer, 100 );
gmo.setOptions( MQGMO_ACCEPT_TRUNCATED_MSG );
queue.get( msg, gmo );

delete [ ] pszBuffer ;
```

V tomto fragmentu kódu lze vyrovnávací paměť vždy adresovat přímo, pomocí *pszBuffer*, na rozdíl od použití metody **bufferPointer** . Je však lepší použít metodu **dataPointer** pro přístup generálního účelu. Aplikace (nikoli objekt třídy ImqCache) musí vyřadit uživatelsky definovanou (neautomatizovanou) vyrovnávací paměť.

Upozornění: Určení ukazatele Null a nulové délky s parametrem **useEmptyBuffer** neurčí pevnou délku vyrovnávací paměti o délce nula, jak by se dalo očekávat. Tato kombinace je interpretována jako požadavek na ignorování jakékoli předchozí vyrovnávací paměti definované uživatelem, a namísto toho se vrátí k použití automatické vyrovnávací paměti.

Zápis zprávy do fronty dead-letter v jazyce C++

Příklad kódu programu pro zápis zprávy do fronty nedoručených zpráv.

Typickým případem zprávy s více částmi je jedna z nich obsahující záhlaví s dead-letter. Data ze zprávy, která nelze zpracovat, se připojí k záhlaví s dead-letter.

```
ImqQueueManager mgr ;        // The queue manager.
ImqQueue queueIn ;           // Incoming message queue.
ImqQueue queueDead ;         // Dead-letter message queue.
ImqMessage msg ;             // Incoming and outgoing message.
ImqDeadLetterHeader header ; // Dead-letter header information.

// Retrieve the message to be rerouted.
queueIn.setConnectionReference( mgr );
queueIn.setName( MY_QUEUE );
queueIn.get( msg );

// Set up the dead-letter header information.
header.setDestinationQueueManagerName( mgr.name( ) );
header.setDestinationQueueName( queueIn.name( ) );
header.setPutApplicationName( /* ? */ );
header.setPutApplicationType( /* ? */ );
header.setPutDate( /* TODAY */ );
header.setPutTime( /* NOW */ );
header.setDeadLetterReasonCode( FB_APPL_ERROR_1234 );

// Insert the dead-letter header information. This will vary
// the encoding, character set and format of the message.
// Message data is moved along, past the header.
msg.writeItem( header );

// Send the message to the dead-letter queue.
```

```
queueDead.setConnectionReference( mgr );
queueDead.setName( mgr.deadLetterQueueName( ) );
queueDead.put( msg );
```

Zápis zprávy do mostu IMS v jazyce C++

Vzorový kód programu pro zápis zprávy na most IMS .

Zprávy odeslané do mostu produktu WebSphere MQ-IMS mohou používat speciální záhlaví. Záhlaví mostu IMS má jako předponu předponu pravidelných dat zprávy.

```
ImqQueueManager    mgr;         // The queue manager.
ImqQueue           queueBridge; // IMS bridge message queue.
ImqMessage         msg;         // Outgoing message.
ImqIMSBridgeHeader header;      // IMS bridge header.

// Set up the message.
//
// Here we are constructing a message with format
// MQFMT_IMS_VAR_STRING, and appropriate data.
//
msg.write( 2,       /* ? */ );         // Total message length.
msg.write( 2,       /* ? */ );         // IMS flags.
msg.write( 7,       /* ? */ );         // Transaction code.
msg.write( /* ? */, /* ? */ );         // String data.
msg.setFormat( MQFMT_IMS_VAR_STRING ); // The format attribute.

// Set up the IMS bridge header information.
//
// The reply-to-format is often specified.
// Other attributes can be specified, but all have default values.
//
header.setReplyToFormat( /* ? */ );

// Insert the IMS bridge header into the message.
//
// This will:
// 1) Insert the header into the message buffer, before the existing
//    data.
// 2) Copy attributes out of the message descriptor into the header,
//    for example the IMS bridge header format attribute will now
//    be set to MQFMT_IMS_VAR_STRING.
// 3) Set up the message attributes to describe the header, in
//    particular setting the message format to MQFMT_IMS.
//
msg.writeItem( header );

// Send the message to the IMS bridge queue.
//
queueBridge.setConnectionReference( mgr );
queueBridge.setName( /* ? */ );
queueBridge.put( msg );
```

Zápis zprávy do mostu CICS v jazyce C++

Vzorový kód programu pro zápis zprávy na most CICS .

Zprávy odeslané do produktu WebSphere MQ pro systém z/OS pomocí mostu CICS vyžadují speciální záhlaví. Hlavička mostu CICS má jako předponu předponu pravidelných dat zprávy.

```
ImqQueueManager mgr ;         // The queue manager.
ImqQueue queueIn ;            // Incoming message queue.
ImqQueue queueBridge ;        // CICS bridge message queue.
ImqMessage msg ;              // Incoming and outgoing message.
ImqCicsBridgeHeader header ;  // CICS bridge header information.

// Retrieve the message to be forwarded.
queueIn.setConnectionReference( mgr );
queueIn.setName( MY_QUEUE );
queueIn.get( msg );

// Set up the CICS bridge header information.
```

```
// The reply-to format is often specified.
// Other attributes can be specified, but all have default values.
header.setReplyToFormat( /* ? */ );

// Insert the CICS bridge header information.  This will vary
// the encoding, character set and format of the message.
// Message data is moved along, past the header.
msg.writeItem( header );

// Send the message to the CICS bridge queue.
queueBridge.setConnectionReference( mgr );
queueBridge.setName( /* ? */ );
queueBridge.put( msg );
```

Psaní zprávy se záhlavím pro práci v jazyce C++

Vzorový kód programu pro zápis zprávy určené pro frontu spravovanou serverem z/OS Workload Manager.

Zprávy odeslané do produktu WebSphere MQ pro z/OS, které jsou určeny pro frontu spravovanou produktem z/OS Workload Manager, vyžadují speciální záhlaví. Hlavička práce je předřazeno k regulárním datům zprávy.

```
ImqQueueManager mgr ;             // The queue manager.
ImqQueue queueIn ;                // Incoming message queue.
ImqQueue queueWLM ;               // WLM managed queue.
ImqMessage msg ;                  // Incoming and outgoing message.
ImqWorkHeader header ;            // Work header information

// Retrieve the message to be forwarded.
queueIn.setConnectionReference( mgr );
queueIn.setName( MY_QUEUE );
queueIn.get( msg );

// Insert the Work header information.  This will vary
// the encoding, character set and format of the message.
// Message data is moved along, past the header.
msg.writeItem( header );

// Send the message to the WLM managed queue.
queueWLM.setConnectionReference( mgr );
queueWLM.setName( /* ? */ );
queueWLM.put( msg );
```

Sestavování programů jazyka C++ produktu WebSphere MQ

Adresa URL podporovaných kompilátorů je uvedena spolu s příkazy, které se používají ke kompilaci, propojení a spuštění programů C++ a ukázek na platformách WebSphere MQ .

Kompilátory pro každou podporovanou platformu a verzi produktu WebSphere MQ jsou uvedeny na stránce systémových požadavků produktu WebSphere MQ na adrese Systémové požadavky pro produkt IBM WebSphere MQ.

Příkaz, který potřebujete zkompilovat a propojit váš program WebSphere MQ C++ závisí na vaší instalaci a požadavcích. Příklady, které následují, ukazují typické kompilační a spojovací příkazy pro některé kompilátory s použitím výchozí instalace produktu WebSphere MQ na různých platformách.

Sestavování programů C++ v systému AIX

Sestavte programy jazyka C++ produktu WebSphere MQ v systému AIX pomocí kompilátoru XL C Enterprise Edition .

Klient

Produkt *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

32bitová aplikace bez podprocesů

```
xlC -o imqsputc_32 imqsput.cpp -qchars=signed -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -limqc23ia -limqb23ia -lmqic
```

32bitovou aplikaci s podprocesy

```
xlC_r -o imqsputc_32_r imqsput.cpp -qchars=signed -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -limqc23ia_r -limqb23ia_r -lmqic_r
```

64bitová aplikace bez podprocesů

```
xlC -q64 -o imqsputc_64 imqsput.cpp -qchars=signed -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -limqc23ia -limqb23ia -lmqic
```

Aplikace s 64bitovým podprocesem

```
xlC_r -q64 -o imqsputc_64_r imqsput.cpp -qchars=signed -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -limqc23ia_r -limqb23ia_r -lmqic_r
```

Server

Produkt *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

32bitová aplikace bez podprocesů

```
xlC -o imqsput_32 imqsput.cpp -qchars=signed -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -limqs23ia -limqb23ia -lmqm
```

32bitovou aplikaci s podprocesy

```
xlC_r -o imqsput_32_r imqsput.cpp -qchars=signed -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -limqs23ia_r -limqb23ia_r -lmqm_r
```

64bitová aplikace bez podprocesů

```
xlC -q64 -o imqsput_64 imqsput.cpp -qchars=signed -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -limqs23ia -limqb23ia -lmqm
```

Aplikace s 64bitovým podprocesem

```
xlC_r -q64 -o imqsput_64_r imqsput.cpp -qchars=signed -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -limqs23ia_r -limqb23ia_r -lmqm_r
```

Sestavování programů C++ v systému HP-UX

Sestavte programy jazyka C++ produktu WebSphere MQ v systému HP-UX pomocí kompilátorů aC+ + nebo aCC .

V systému HP-UX Itaniumpodporuje produkt WebSphere MQ pouze standardní běhový modul. Použijte kompilátor aCC .

- Soubor libimqi23bh.sl poskytuje třídy jazyka C++ produktu WebSphere MQ pro standardní běhový modul.
- Z důvodu kompatibility s dřívějšími verzemi je poskytnut symbolický odkaz z knihovny libimqi23ah.sl do souboru libimqi23bh.sl.

IA64 (IPF)

Produkt *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, v němž je nainstalován produkt WebSphere MQ .

Klient: IA64 (IPF)

32bitová aplikace bez podprocesů

```
aCC -Wl,+b,: +e -D_HPUX_SOURCE -o imqsputc_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -L/usr/lib/hpux32 -limqi23bh -lmqic
```

32bitovou aplikaci s podprocesy

```
aCC -Wl,+b,: +e -D_HPUX_SOURCE -o imqsputc_32_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -L/usr/lib/hpux32 -limqi23bh_r -lmqic_r -lpthread
```

64bitová aplikace bez podprocesů

```
aCC +DD64 +e -D_HPUX_SOURCE -o imqsputc_64 imqsput.cpp
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -limqi23bh -lmqic
```

Aplikace s 64bitovým podprocesem

```
aCC +DD64 +e -D_HPUX_SOURCE -o imqsputc_64_r imqsput.cpp
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -limqi23bh_r
-lmqic_r
-lpthread
```

Server: IA64 (IPF)

32bitová aplikace bez podprocesů

```
aCC -Wl,+b,: +e -D_HPUX_SOURCE -o imqsput_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -L/usr/lib/hpux32 -limqi23bh -lmqm
```

32bitovou aplikaci s podprocesy

```
aCC -Wl,+b,: +e -D_HPUX_SOURCE -o imqsput_32_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -L/usr/lib/hpux32 -limqi23bh_r -lmqm_r -lpthread
```

64bitová aplikace bez podprocesů

```
aCC +DD64 +e -D_HPUX_SOURCE -o imqsput_64 imqsput.cpp
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -limqi23bh -lmqm
```

Aplikace s 64bitovým podprocesem

```
aCC +DD64 +e -D_HPUX_SOURCE -o imqsput_64_r imqsput.cpp
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -limqi23bh_r
-lmqm_r
-lpthread
```

Sestavování programů C++ v systému Linux

Sestavte programy jazyka C++ produktu WebSphere MQ v systému Linux pomocí kompilátoru GNU g + +.

Systém p

Produkt *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

Klient: System p

32bitová aplikace bez podprocesů

```
g++ -m32 -o imqsputc_32 imqsput.cpp -fsigned-char -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqc23gl
-limqb23gl -lmqic
```

32bitovou aplikaci s podprocesy

```
g++ -m32 -o imqsputc_r32 imqsput.cpp -fsigned-char -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqc23gl_r
-limqb23gl_r -lmqic_r
```

64bitová aplikace bez podprocesů

```
g++ -m64 -o imqsputc_64 imqsput.cpp -fsigned-char -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqc23gl -limqb23gl -lmqic
```

Aplikace s 64bitovým podprocesem

```
g++ -m64 -o imqsputc_r64 imqsput.cpp -fsigned-char -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqc23gl_r -limqb23gl_r -lmqic_r
```

Server: System p

32bitová aplikace bez podprocesů

```
g++ -m32 -o imqsput_32 imqsput.cpp -fsigned-char -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqs23gl
-limqb23gl -lmqm
```

32bitovou aplikaci s podprocesy

```
g++ -m32 -o imqsput_r32 imqsput.cpp -fsigned-char -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqs23gl_r
-limqb23gl_r -lmqm_r
```

64bitová aplikace bez podprocesů

```
g++ -m64 -o imqsput_64 imqsput.cpp -fsigned-char -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqs23gl -limqb23gl -lmqm
```

Aplikace s 64bitovým podprocesem

```
g++ -m64 -o imqsput_r64 imqsput.cpp -fsigned-char -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqs23gl_r -limqb23gl_r -lmqm_r
```

System z

Produkt *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

Klient: System z

32bitová aplikace bez podprocesů

```
g++ -m31 -fsigned-char -o imqsputc_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqc23gl -limqb23gl -lmqic
```

32bitovou aplikaci s podprocesy

```
g++ -m31 -fsigned-char -o imqsputc_32_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqc23gl_r -limqb23gl_r -lmqic_r
-lpthread
```

64bitová aplikace bez podprocesů

```
g++ -m64 -fsigned-char -o imqsputc_64 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqc23gl -limqb23gl -lmqic
```

Aplikace s 64bitovým podprocesem

```
g++ -m64 -fsigned-char -o imqsputc_64_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64  -Wl,-rpath=/usr/lib64
-limqc23gl_r -limqb23gl_r -lmqic_r -lpthread
```

Server: System z

32bitová aplikace bez podprocesů

```
g++ -m31 -fsigned-char -o imqsput_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib  -Wl,-rpath=/usr/lib
-limqs23gl -limqb23gl -lmqm
```

32bitovou aplikaci s podprocesy

```
g++ -m31 -fsigned-char -o imqsput_32_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqs23gl_r -limqb23gl_r -lmqm_r -lpthread
```

64bitová aplikace bez podprocesů

```
g++ -m64 -fsigned-char -o imqsput_64 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqs23gl -limqb23gl -lmqm
```

Aplikace s 64bitovým podprocesem

```
g++ -m64 -fsigned-char -o imqsput_64_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqs23gl_r -limqb23gl_r -lmqm_r -lpthread
```

System x (32bitový)

Produkt *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

Klient: System x (32bitový)

32bitová aplikace bez podprocesů

```
g++ -m32 -fsigned-char -o imqsputc_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -LMQ_INSTALLATION_PATH/lib
-Wl,
-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -limqc23gl -limqb23gl -lmqic
```

32bitovou aplikaci s podprocesy

```
g++ -m32 -fsigned-char -o imqsputc_32_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -LMQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -limqc23gl_r -limqb23gl_r
-lmqic_r -lpthread
```

64bitová aplikace bez podprocesů

```
g++ -m64 -fsigned-char -o imqsputc_64 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -LMQ_INSTALLATION_PATH/
lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -limqc23gl -limqb23gl
-lmqic
```

Aplikace s 64bitovým podprocesem

```
g++ -m64 -fsigned-char -o imqsputc_64_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -LMQ_INSTALLATION_PATH/
lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -limqc23gl_r -limqb23gl_r
-lmqic_r -lpthread
```

Server: System x (32bitový)

32bitová aplikace bez podprocesů

```
g++ -m32 -fsigned-char -o imqsput_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -LMQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -limqs23gl -limqb23gl -lmqm
```

32bitovou aplikaci s podprocesy

```
g++ -m32 -fsigned-char -o imqsput_32_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATHlib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -LMQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -limqs23gl_r -limqb23gl_r
-lmqm_r -lpthread
```

64bitová aplikace bez podprocesů

```
g++ -m64 -fsigned-char -o imqsput_64 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -LMQ_INSTALLATION_PATH/
lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -limqs23gl -limqb23gl -lmqm
```

Aplikace s 64bitovým podprocesem

```
g++ -m64 -fsigned-char -o imqsput_64_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -LMQ_INSTALLATION_PATH/
lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -limqs23gl_r -limqb23gl_r
-lmqm_r -lpthread
```

Sestavování programů C++ v systému Solaris

Sestavte programy jazyka C++ produktu WebSphere MQ v systému Solaris pomocí kompilátoru Sun ONE.

SPARC

Produkt *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

Klient: SPARC

32bitová aplikace

```
CC -xarch=v8plus -mt -o imqsputc_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -RMQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqc23as -limqb23as
-lmqic -lsocket -lnsl -ldl
```

64bitové aplikace

```
CC -xarch=v9 -mt -o imqsputc_64 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqc23as
-limqb23as
-lmqic -lsocket -lnsl -ldl
```

Server: SPARC

32bitová aplikace

```
CC -xarch=v8plus -mt -o imqsput_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -RMQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqs23as -limqb23as
-lmqm -lsocket -lnsl -ldl
```

64bitové aplikace

```
CC -xarch=v9 -mt -o imqsput_64 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqs23as
-limqb23as
-lmqm -lsocket -lnsl -ldl
```

x86-64

Produkt *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

Klient: x86-64

32bitová aplikace

```
CC -xarch=386 -mt -o imqsputc_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -RMQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqc23as -limqb23as
-lmqic -lsocket -lnsl -ldl
```

64bitové aplikace

```
CC -xarch=amd64 -mt -o imqsputc_64 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqc23as
-limqb23as
-lmqic -lsocket -lnsl -ldl
```

Server: x86-64

32bitová aplikace

```
CC -xarch=386 -mt -o imqsput_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
```

```
-LMQ_INSTALLATION_PATH/lib -RMQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqs23as -limqb23as
-lmqm -lsocket -lnsl -ldl
```

64bitové aplikace

```
CC -xarch=amd64 -mt -o imqsput_64 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqs23as
-limqb23as
-lmqm -lsocket -lnsl -ldl
```

Sestavování programů C++ v systému Windows

Sestavte programy WebSphere MQ C++ v systému Windows pomocí kompilátoru jazyka Microsoft Visual Studio C + +.

Soubory knihovny (.lib) a soubory dll pro použití s 32bitovými aplikacemi jsou nainstalovány v produktu *MQ_INSTALLATION_PATH*/Tools/Lib, soubory pro použití s 64bitovými aplikacemi jsou nainstalovány v produktu *MQ_INSTALLATION_PATH*/Tools/Lib64. *MQ_INSTALLATION_PATH* Představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

Klient

```
cl -MD imqsput.cpp /Feimqsputc.exe imqb23vn.lib imqc23vn.lib
```

Server

```
 cl -MD imqsput.cpp /Feimqsput.exe imqb23vn.lib imqs23vn.lib
```

Použití tříd produktu WebSphere MQ pro prostředí Java

Třídy WebSphere MQ pro prostředí Java umožňují používat produkt WebSphere MQ v prostředí Java. A Java application can use either WebSphere MQ classes for Java or WebSphere MQ classes for JMS to access WebSphere MQ resources.

Třídy WebSphere MQ for Java umožňují aplikaci Java:

- Připojte se k produktu WebSphere MQ jako klienta WebSphere MQ
- Připojit přímo ke správci front WebSphere MQ

Třídy WebSphere MQ pro prostředí Java zapouzdřují rozhraní MQI (Message Queue Interface), nativní rozhraní API produktu WebSphere MQ .

Třídy WebSphere MQ pro jazyk Java používají podobný model objektu pro rozhraní jazyka C++ a .NET pro produkt WebSphere MQ.

Proč bych měl používat třídy WebSphere MQ pro prostředí Java?

Pokud jsou ve vaší instalaci významné následující body, zvažte použití tříd produktu WebSphere MQ pro prostředí Java:

- Třídy WebSphere MQ pro prostředí Java zapouzdřují rozhraní MQI (Message Queue Interface), nativní rozhraní API produktu WebSphere MQ .
  - Pokud jste obeznámeni s použitím rozhraní MQI v procedurálních jazycích, můžete tyto znalosti přenést do prostředí Java.
  - Můžete využít celou řadu funkcí produktu WebSphere MQnad rámec těch, které jsou k dispozici prostřednictvím rozhraní JMS.

- Třídy WebSphere MQ pro jazyk Java používají podobný model objektu pro rozhraní jazyka C++ a .NET pro produkt WebSphere MQ. Pokud jste obeznámeni s těmito rozhraními, můžete tyto znalosti přenést do prostředí Java.

Poznámka: Automatické opětovné připojení klienta není podporováno třídami produktu WebSphere MQ pro prostředí Java.

Začínáme s třídami produktu WebSphere MQ pro prostředí Java

Tato kolekce témat poskytuje přehled tříd produktu WebSphere MQ pro jazyk Java a jejich použití.

Co jsou třídy produktu WebSphere MQ pro prostředí Java?

Třídy WebSphere MQ for Java umožňují používat produkt WebSphere MQ v prostředí Java.

Třídy WebSphere MQ for Java umožňují aplikaci Java:

- Připojte se k produktu WebSphere MQ jako klienta WebSphere MQ
- Připojit přímo ke správci front WebSphere MQ

Třídy WebSphere MQ pro prostředí Java zapouzdřují rozhraní MQI (Message Queue Interface), nativní rozhraní API produktu WebSphere MQ .

Třídy WebSphere MQ pro jazyk Java používají podobný model objektu pro rozhraní jazyka C++ a .NET pro produkt WebSphere MQ.

Proč bych měl používat třídy WebSphere MQ pro prostředí Java?

A Java application can use either WebSphere MQ classes for Java or WebSphere MQ classes for JMS to access WebSphere MQ resources. Pro použití tříd produktu WebSphere MQ pro jazyk Java je k dispozici řada výhod.

Pokud jsou ve vaší instalaci významné následující body, zvažte použití tříd Websphere MQ pro jazyk Java:

- Třídy WebSphere MQ pro prostředí Java zapouzdřují rozhraní MQI (Message Queue Interface), nativní rozhraní API produktu WebSphere MQ .
  - Pokud jste obeznámeni s použitím rozhraní MQI v procedurálních jazycích, můžete tyto znalosti přenést do prostředí Java.
  - Můžete využít celou řadu funkcí produktu WebSphere MQnad rámec těch, které jsou k dispozici prostřednictvím rozhraní JMS.
- Třídy WebSphere MQ pro jazyk Java používají podobný model objektu pro rozhraní jazyka C++ a .NET pro produkt WebSphere MQ. Pokud jste obeznámeni s těmito rozhraními, můžete tyto znalosti přenést do prostředí Java.

Volby připojení pro třídy WebSphere MQ pro prostředí Java

Třídy WebSphere MQ pro jazyk Java se mohou připojit v režimu klienta nebo vazby.

Programovatelné volby umožňují produktu WebSphere MQ Classes for Java připojit se k produktu WebSphere MQ jedním z následujících způsobů:

- Jako klient WebSphere MQ MQI pomocí protokolu Transmission Control Protocol/Internet Protocol (TCP/IP)
- V režimu vázání se připojuje přímo k produktu WebSphere MQ pomocí rozhraní JNI (Java Native Interface).

Klienty nelze spustit v systému z/OS, ale klienti na jiných platformách se mohou připojit ke správci front WebSphere MQ pro správce front z/OS , je-li nainstalován produkt Client Attach Facility.

Následující části popisují volby připojení režimu klienta a režimu vazeb podrobněji.

Připojení klienta

Chcete-li se připojit ke správci front v režimu klienta, mohou být třídy produktu WebSphere MQ pro aplikaci Java spuštěny ve stejném systému, v němž je spuštěn správce front, nebo na jiném systému. V každém případě se třídy produktu WebSphere MQ pro prostředí Java připojují ke správci front prostřednictvím protokolu TCP/IP.

Třídy produktu WebSphere MQ pro aplikaci Java se mohou připojit k libovolnému podporovanému správci front s použitím režimu klienta.

Další informace o tom, jak zapisovat aplikace pro použití připojení v režimu klienta, najdete v tématu "Režimy připojení WebSphere MQ pro režimy připojení Java" na stránce 646.

Připojení vazeb

Při použití v režimu vázání používá produkt WebSphere MQ Classes for Java rozhraní JNI (Java Native Interface) k volání přímo do existujícího rozhraní API správce front, a nikoli prostřednictvím komunikace prostřednictvím sítě. Ve většině prostředí poskytuje připojení v režimu vazeb lepší výkon pro třídy WebSphere MQ pro aplikace Java, než se připojuje v režimu klienta, a to tak, že se vyhnete nákladům na komunikaci TCP/IP.

Aplikace, které používají třídy WebSphere MQ pro prostředí Java pro připojení v režimu vazeb, musí být spuštěny ve stejném systému jako správce front, ke kterému se připojují.

Běhové prostředí Java Runtime Environment, které se používá ke spouštění tříd produktu WebSphere MQ pro aplikaci Java, musí být konfigurováno k načtení tříd produktu WebSphere MQ pro knihovny Java; další informace naleznete v tématu Třídy WebSphere MQ pro knihovny Java .

Další informace o tom, jak zapisovat aplikace pro použití připojení v režimu vázání, viz "Režimy připojení WebSphere MQ pro režimy připojení Java" na stránce 646.

Nezbytné předpoklady pro třídy WebSphere MQ pro prostředí Java

Chcete-li používat třídy WebSphere MQ pro prostředí Java, potřebujete některé další softwarové produkty.

Nejnovější informace o předpokladech pro třídy WebSphere MQ pro jazyk Java naleznete v souboru Readme produktu WebSphere MQ .

Chcete-li vyvíjet třídy WebSphere MQ pro aplikace v jazyce Java, potřebujete sadu JDK (Java Development Kit). Podrobnosti sad JDK, které jsou podporovány vaším operačním systémem, najdete na stránce systémových požadavků produktu WebSphere MQ na adrese Systémové požadavky pro produkt IBM WebSphere MQ.

Chcete-li spustit třídy WebSphere MQ pro aplikace Java, potřebujete tyto softwarové komponenty:

- Správce front produktu WebSphere MQ pro aplikace, které se připojují ke správci front
- Běhové prostředí Java Runtime Environment (JRE) pro každý systém, na kterém spouštíte aplikace. Vhodné prostředí JRE je dodáváno s produktem WebSphere MQ.

Požadujete-li připojení SSL k použití šifrovacích modulů, které byly certifikovány FIPS 140-2, potřebujete poskytovatele IBM Java JSSE FIPS (IBMJSSEFIPS). Každá sada IBM JDK a prostředí JRE ve verzi 1.4.2 nebo novější obsahuje IBMJSSEFIPS.

Můžete použít adresy Internet Protocol verze 6 (IPv6) ve vašich třídách WebSphere MQ pro aplikace Java, pokud IPv6 je podporovaná vaším prostředím JVM (Java Virtual Machine) a implementací TCP/IP ve vašem operačním systému.

Instalace a konfigurace tříd produktu WebSphere MQ pro prostředí Java

Tato sekce popisuje adresáře a soubory vytvořené při instalaci tříd produktu WebSphere MQ pro jazyk Java a informuje o tom, jak nakonfigurovat třídy produktu WebSphere MQ pro prostředí Java po instalaci.

Co je nainstalováno pro třídy WebSphere MQ pro prostředí Java

Nejnovější verze tříd produktu WebSphere MQ pro jazyk Java je nainstalována s produktem WebSphere MQ. Je možné, že budete muset přepsat výchozí volby instalace, abyste se ujistili, že je to hotovo.

Další informace o instalaci produktu WebSphere MQ naleznete v následujících tématech:

Instalace serveru WebSphere MQ
Instalace klienta IBM WebSphere MQ

Třídy WebSphere MQ pro prostředí Java jsou obsaženy v souborech archivu Java (JAR), com.ibm.mq.jara com.ibm.mq.jmqi.jar.

Podpora pro standardní záhlaví zpráv, jako je Programmable Command Format (PCF), je obsažena v souboru JAR com.ibm.mq.headers.jar.

Podpora pro Programmable Command Format (PCF) je obsažena v souboru JAR com.ibm.mq.pcf.jar.

Instalace a upgrade tříd produktu WebSphere MQ pro soubory JAR Java

Jediným podporovaným způsobem získání tříd WebSphere MQ pro soubory JAR prostředí Java v systému je instalace produktu WebSphere MQ nebo klienta WebSphere MQ MQI SupportPacnebo použití nástroje pro správu softwaru, jako např. Apache Maven, kde naleznete další informace o produktu “IBM WebSphere MQ classes for Java a nástroje pro správu softwaru” na stránce 642.

Nepřesunujte ani nekopírujte třídy WebSphere MQ pro soubory JAR prostředí Java z jiných počítačů, pokud nepoužíváte nástroj pro správu softwaru.

- Opravné sady nelze aplikovat na "instalaci", kde byly soubory JAR zkopírovány z jiného počítače a je mnohem těžší zajistit, aby všechny soubory JAR byly udržovány v jednotlivých krocích, a jsou na kompatibilních úrovních.
- Zkopírování tříd produktu WebSphere MQ pro soubory JAR platformy JMS mezi počítači může také vést k více kopiím souborů umístěných na stejném počítači, což může způsobit problémy při opravě kódu a ladění problémů.

Do archivů aplikací nezahrnujte třídy WebSphere MQ pro soubory JAR Java.

- Aktualizace tříd produktu WebSphere MQ pro prostředí Java nelze použít v produktu WebSphere MQ Fix Pack.
- Podpora IBM nepodporuje možnost snadného určení verze tříd WebSphere MQ pro jazyk Java, které aplikace používá.
- Problémy mohou nastat, pokud více aplikací spuštěných ve stejném běhovém prostředí Java zahrnuje různé verze tříd produktu WebSphere MQ pro jazyk Java, protože více verzí tříd WebSphere MQ pro jazyk Java je načíteno do prostředí Java Runtime Environment současně.
- Pokud aplikace používá přenos BINDINGS pro připojení ke správci front, všechny hlavní přechody na vyšší verzi správce front rovněž vyžadují aktualizaci aplikace tak, aby zahrnovala odpovídající úroveň tříd WebSphere MQ pro prostředí Java.

  Je-li například správce front upgradován na úroveň produktu WebSphere MQ verze 7.1 , musí být všechny aplikace, které se připojují ke správci front pomocí přenosu BINDINGS, aktualizovány také tak, aby zahrnovaly třídy WebSphere MQ verze 7.1 pro jazyk Java.

Následující knihovna Java je distribuována s třídami WebSphere MQ pro jazyk Java:

- connector.jar (verze 1.0)

Ukázková aplikace s názvem Pohlednice se nachází v souboru JAR com.ibm.mq.postcard.jar.

Nástroj Javadoc se používá ke generování stránek HTML obsahujících specifikace tříd WebSphere MQ pro třídy Java a WebSphere MQ pro rozhraní JMS API. Stránky HTML jsou umístěny v podadresáři doc adresáře WebSphere MQ pro instalační adresář platformy JMS. Na systémech UNIX, Linuxa Windows obsahuje podadresář doc jednotlivé stránky HTML.

Po dokončení instalace jsou soubory a ukázky nainstalovány v umístěních zobrazených v produktu “Instalační adresáře pro třídy produktu WebSphere MQ pro prostředí Java” na stránce 635.

Po instalaci je třeba aktualizovat proměnné prostředí na jiné platformě než Windowstak, jak je popsáno v tématu “Proměnné prostředí relevantní pro třídy produktu WebSphere MQ pro prostředí Java” na stránce 635.

Instalační adresáře pro třídy produktu WebSphere MQ pro prostředí Java

Soubory třídy WebSphere MQ pro soubory Java jsou nainstalovány v různých umístěních podle platformy.

Tabulka 82 na stránce 635 zobrazuje, kde jsou nainstalovány třídy WebSphere MQ pro soubory Java.

Tabulka 82. Třídy WebSphere MQ pro instalační adresáře Java

Platforma	**Adresář**
AIX	*MQ_INSTALLATION_PATH*/java/lib
HP-UX, Linuxa Solaris	*MQ_INSTALLATION_PATH*/java/lib
Windows	*MQ_INSTALLATION_PATH*\java\lib
Produkt *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.	

Některé ukázkové aplikace, jako např. Ověřovací programy instalace (IVP), jsou dodávány s produktem WebSphere MQ. Tabulka 83 na stránce 635 ukazuje, kde jsou ukázkové aplikace nainstalovány. Třídy WebSphere MQ pro ukázky jazyka Java jsou umístěny v podadresáři s názvem `wmqjava`. Ukázky PCF se nacházejí v podadresáři s názvem `pcf`.

Tabulka 83. Adresáře ukázek

Platforma	**Adresář**
AIX	*MQ_INSTALLATION_PATH*/samp/wmqjava/
HP-UX, Linuxa Solaris	*MQ_INSTALLATION_PATH*/samp/wmqjava/
Windows	*MQ_INSTALLATION_PATH*\tools\wmqjava\
Produkt *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.	

Proměnné prostředí relevantní pro třídy produktu WebSphere MQ pro prostředí Java

Chcete-li spustit třídy WebSphere MQ pro aplikace v jazyce Java, musí jejich cesty ke třídám obsahovat třídy WebSphere MQ pro adresáře Java a ukázky.

Aby bylo možné spustit třídy WebSphere MQ pro aplikace v jazyce Java, musí cesta ke třídám obsahovat příslušné třídy WebSphere MQ pro adresář Java. Chcete-li spustit ukázkové aplikace, musí cesta ke třídám obsahovat také příslušné adresáře ukázek. Tyto informace mohou být poskytnuty v příkazu vyvolání Java nebo v proměnné prostředí CLASSPATH.

Tabulka 84 na stránce 635 zobrazuje příslušnou hodnotu CLASSPATH, která má být použita na každé platformě ke spuštění tříd produktu WebSphere MQ pro aplikace v jazyce Java, včetně ukázkových aplikací.

Tabulka 84. CLASSPATH nastavení pro spuštění tříd WebSphere MQ pro aplikace v jazyce Java, včetně tříd WebSphere MQ pro ukázkové aplikace v jazyce Java

Platforma	**Nastavení CLASSPATH**
AIX	CLASSPATH=*MQ_INSTALLATION_PATH*/java/lib/com.ibm.mq.jar: *MQ_INSTALLATION_PATH*/samp/wmqjava/samples:

Tabulka 84. CLASSPATH nastavení pro spuštění tříd WebSphere MQ pro aplikace v jazyce Java, včetně tříd WebSphere MQ pro ukázkové aplikace v jazyce Java (pokračování)	
Platforma	**Nastavení CLASSPATH**
HP-UX, Linuxa Solaris	CLASSPATH=*MQ_INSTALLATION_PATH*/java/lib/com.ibm.mq.jar: *MQ_INSTALLATION_PATH*/samp/wmqjava/samples:
Windows	CLASSPATH=*MQ_INSTALLATION_PATH*\Java\lib\com.ibm.mq.jar; *MQ_INSTALLATION_PATH*\tools\wmqjava\samples;
Produkt *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.	

Při kompilaci pomocí volby -Xlint se může zobrazit zpráva s varováním, že soubor com.ibm.mq.ese.jar není přítomen. Varování můžete ignorovat. Tento soubor je přítomen pouze v případě, že jste nainstalovali produkt IBM WebSphere MQ Advanced Message Security.

Skripty poskytnuté s třídami produktu WebSphere MQ pro prostředí Java používají následující proměnné prostředí:

MQ_JAVA_DATA_PATH
Tato proměnná prostředí určuje adresář pro výstup protokolu a trasování.

INSTALAČNÍ_CESTA MQ_JAVA_INSTALL_PATH
Tato proměnná prostředí určuje adresář, kde jsou instalovány třídy WebSphere MQ pro Javu, jak je zobrazeno v WebSphere MQ classes for Java installation directories.

KOŘEN ROZHRANÍ MQ_JAVA_LIB_PATH
Tato proměnná prostředí určuje adresář, kde jsou uloženy třídy produktu WebSphere MQ pro knihovny Java, jak ukazuje Umístění tříd produktu WebSphere MQ pro knihovny Java pro každou platformu. Některé skripty dodávané s třídami produktu WebSphere MQ pro prostředí Java, jako např. IVTRun, používají tuto proměnnou prostředí.

V systému Windowsjsou všechny proměnné prostředí nastaveny automaticky během instalace. Na jakékoliv jiné platformě je musíte nastavit sami.Na systému UNIX můžete použít skript **setjmsenv** (používáte-li 32bitové prostředí JVM) nebo **setjmsenv64** (pokud používáte 64bitové prostředí JVM) k nastavení proměnných prostředí. V systému AIX, HP-UX, Linuxa Solaris se tyto skripty nacházejí v adresáři *MQ_INSTALLATION_PATH*/java/bin .

Třídy IBM WebSphere MQ pro knihovny Java

Umístění tříd produktu IBM WebSphere MQ pro knihovny Java se liší v závislosti na platformě. Uveďte toto umístění, když spustíte aplikaci.

Chcete-li zadat umístění knihoven JNI (Java Native Interface), spusťte aplikaci pomocí příkazu **java** s následujícím formátem:

```
java -Djava.library.path=library_path application_name
```

kde *cesta_knihovny* je cesta k třídám produktu WebSphere MQ pro knihovny Java, které zahrnují knihovny JNI. Tabulka 85 na stránce 637 Zobrazuje umístění tříd produktu WebSphere MQ pro knihovny Java pro každou platformu.

Tabulka 85. Umístění tříd produktu WebSphere MQ pro knihovny Java pro každou platformu	
Platforma	**Adresář obsahující třídy WebSphere MQ pro knihovny Java**
AIX	*MQ_INSTALLATION_PATH*/java/lib (32bitové knihovny) *MQ_INSTALLATION_PATH*/java/lib64 (64 bitové knihovny)
HP-UX Linux (POWER, x86-64 a zSeries s390x platformy) Solaris (platformyx86-64 a SPARC)	*MQ_INSTALLATION_PATH*/java/lib (32bitové knihovny) *MQ_INSTALLATION_PATH*/java/lib64 (64 bitové knihovny)
Linux (platformax86)	*MQ_INSTALLATION_PATH*/java/bin
Windows	*MQ_INSTALLATION_PATH*\Java\lib (32bitové knihovny) *MQ_INSTALLATION_PATH*\Java\lib64 (64bitové knihovny)
Produkt *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.	

Poznámka:

1. Na systémech AIX, HP-UX, Linux (Power platform) nebo Solaris použijte buď 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. V systému Windowsmůžete použít proměnnou prostředí PATH k určení umístění tříd WebSphere MQ pro knihovny jazyka Java namísto zadávání jejich umístění v příkazu **java** .
3. Chcete-li používat třídy WebSphere MQ pro prostředí Java v režimu vázání v produktu IBM i, ujistěte se, že knihovna QMQMJAVA je ve vašem seznamu knihoven.

Související úlohy

Použití tříd produktu WebSphere MQ pro prostředí Java

Podpora pro OSGi na serveru IBM WebSphere MQ classes for Java

OSGi poskytuje rámec, který podporuje implementaci aplikací jako balíky. Jeden balík OSGi je dodáván jako část produktu IBM WebSphere MQ classes for Java .

OSGi poskytuje obecný účel, zabezpečený a spravovaný rámecJava , který podporuje implementaci aplikací, které jsou dodávány ve formě balíků. Zařízení vyhovující specifikaci OSGi mohou stahovat a instalovat balíky a odebírat je, pokud již nejsou zapotřebí. Rámec spravuje instalaci a aktualizaci balíků v dynamickém a přizpůsobitelném stylu.

IBM WebSphere MQ classes for Java. zahrnuje následující balík OSGi.

com.ibm.mq.osgi.java_< číslo verze > .jar
Soubory JAR, které umožní aplikacím používat produkt IBM WebSphere MQ classes for Java.

kde < číslo verze > je číslo verze produktu WebSphere MQ , který byl nainstalován.

Balík je nainstalován do podadresáře `java/lib/OSGi` vaší instalace produktu IBM WebSphere MQ nebo složky `java\lib\OSGi` v systému Windows.

Devět dalších balíků je také nainstalováno do podadresáře `java/lib/OSGi` vaší instalace produktu IBM WebSphere MQ nebo složky `java\lib\OSGi` v systému Windows. Tyto balíky jsou součástí produktu IBM WebSphere MQ classes for JMS a nesmí být načteny do běhového prostředí OSGi, které má načtený balík IBM WebSphere MQ classes for Java . Je-li balík OSGi produktu IBM WebSphere MQ classes for Java načten do běhového prostředí OSGi, které má také načtené balíky IBM WebSphere MQ classes for JMS , chyby jako jsou:

```
java.lang.ClassCastException: com.ibm.mq.MQException incompatible with com.ibm.mq.MQException
```

Vyskytne se, když se spustí aplikace používající buď balík IBM WebSphere MQ classes for Java , nebo balíky IBM WebSphere MQ classes for JMS .

Svazek balíků OSGi pro produkt IBM WebSphere MQ classes for Java byl zapsán do specifikace OSGi Release 4. Nepracuje v prostředí OSGi Release 3.

Musíte správně nastavit systémovou cestu nebo cestu ke knihovně tak, aby běhové prostředí OSGi mohlo najít všechny požadované soubory DLL nebo sdílené knihovny.

Pokud použijete svazek balíků OSGi pro produkt IBM WebSphere MQ classes for Java, třídy uživatelské procedury kanálu zapsané v produktu Java nejsou podporovány kvůli inherentním problémům při načítání tříd v prostředí zavaděče více tříd, jako je např. OSGi. Balík uživatelů si může být vědom balíku IBM WebSphere MQ classes for Java , ale balík IBM WebSphere MQ classes for Java si není vědom žádného uživatelského balíku. V důsledku toho zavaděč tříd použitý v balíku IBM WebSphere MQ classes for Java nemůže načíst třídu ukončení kanálu, která je v uživatelském balíku.

Další informace o specifikaci OSGi naleznete na webu Alianci OSGi .

Konfigurační soubor IBM WebSphere MQ classes for Java

Konfigurační soubor produktu IBM WebSphere MQ classes for Java uvádí vlastnosti, které se používají ke konfiguraci agenta IBM WebSphere MQ classes for Java.

Formát konfiguračního souboru produktu IBM WebSphere MQ classes for Java je standardní soubor vlastností produktu Java .

V 7.5.0.9 V produktu IBM WebSphere MQ Version 7.5.0, opravná sada Fix Pack 9je ukázkový konfigurační soubor, který se nazývá `mqjava.config` , dodán v podadresáři `bin` instalačního adresáře produktu IBM WebSphere MQ classes for Java . Tento soubor dokumentuje všechny podporované vlastnosti a jejich výchozí hodnoty.

Poznámka: Ukázkový konfigurační soubor je přepsán při upgradu instalace produktu IBM WebSphere MQ na budoucí opravnou sadu Fix Pack. Proto se doporučuje vytvořit kopii ukázkového konfiguračního souboru pro použití s vašimi aplikacemi.

Můžete zvolit název a umístění konfiguračního souboru produktu IBM WebSphere MQ classes for Java . Když spustíte aplikaci, použijte příkaz **java** s následujícím formátem:

```
java -Dcom.ibm.msg.client.config.location=config_file_url application_name
```

V příkazu *config_file_url* je adresa URL (Uniform Resource Locator), která určuje název a umístění konfiguračního souboru produktu IBM WebSphere MQ classes for Java . Podporovány jsou adresy URL následujících typů: `http`, `file`, `ftp`a `jar`.

Následující příklad ukazuje příkaz **java** :

```
java -Dcom.ibm.msg.client.config.location=file:/D:/mydir/mqjava.config MyAppClass
```

Tento příkaz identifikuje konfigurační soubor IBM WebSphere MQ classes for Java jako soubor `D:\mydir\mqjava.config` na lokálním systému Windows .

Konfigurační soubor produktu IBM WebSphere MQ classes for Java lze použít s libovolným z podporovaných přenosů mezi aplikací a správcem front nebo zprostředkovatelem.

Potlačení vlastností uvedených v konfiguračním souboru IBM WebSphere MQ classes for Java

Konfigurační soubor produktu IBM WebSphere MQ MQI client může také určovat vlastnosti, které se používají ke konfiguraci produktu IBM WebSphere MQ classes for Java. Vlastnosti určené v konfiguračním souboru produktu IBM WebSphere MQ MQI client se však používají pouze v případě, že se aplikace připojuje ke správci front v režimu klienta.

Je-li to nutné, můžete přepsat jakýkoli atribut v konfiguračním souboru IBM WebSphere MQ MQI client tak, že jej uvedete jako vlastnost v konfiguračním souboru IBM WebSphere MQ classes for Java . Chcete-li přepsat atribut v konfiguračním souboru IBM WebSphere MQ MQI client , použijte položku s následujícím formátem v konfiguračním souboru IBM WebSphere MQ classes for Java :

```
com.ibm.mq.cfg.stanza.propName=propValue
```

Proměnné v položce mají následující význam:

sekce
Název stanzy v konfiguračním souboru IBM WebSphere MQ MQI client , který obsahuje atribut.

propName
Název atributu, jak je uveden v konfiguračním souboru IBM WebSphere MQ MQI client .

propValue
Hodnota vlastnosti, která přepíše hodnotu atributu, která je uvedena v konfiguračním souboru IBM WebSphere MQ MQI client .

Eventuálně můžete přepsat atribut v konfiguračním souboru IBM WebSphere MQ MQI client určením vlastnosti jako systémové vlastnosti v příkazu **java** . Chcete-li určit vlastnost jako systémovou vlastnost, použijte předchozí formát.

Pouze následující atributy v konfiguračním souboru IBM WebSphere MQ MQI client jsou relevantní pro produkt IBM WebSphere MQ classes for Java. Pokud uvedete nebo přepisují jiné atributy, nemá žádný efekt. Konkrétně si všimněte, že `ChannelDefinitionFile` a `ChannelDefinitionDirectory` v sekci CHANNELS v konfiguračním souboru klienta se nepoužívají. Podrobné informace o použití tabulky CCDT s produktem IBM WebSphere MQ classes for Javaviz “Použití tabulky definic kanálů klienta s IBM WebSphere MQ classes for Java” na stránce 650 .

Tabulka 86. Který oddíl konfiguračního souboru klienta obsahuje který atribut

sekce	**Atribut**
ClientExitSekce Cesta ke konfiguračnímu souboru klienta	ExitsDefaultPath
ClientExitSekce Cesta ke konfiguračnímu souboru klienta	ExitsDefaultPath64
ClientExitSekce Cesta ke konfiguračnímu souboru klienta	Cesta ke třídě JavaExits
stanzaMessageBuffer konfiguračního souboru klienta	MaximumSize
stanzaMessageBuffer konfiguračního souboru klienta	PurgeTime
stanzaMessageBuffer konfiguračního souboru klienta	UpdatePercentage
Sekce TCP konfiguračního souboru klienta	ClntRcvBufSize
Sekce TCP konfiguračního souboru klienta	ClntSndBufSize
Sekce TCP konfiguračního souboru klienta	Časový limit připojení

Tabulka 86. Který oddíl konfiguračního souboru klienta obsahuje který atribut (pokračování)	
sekce	**Atribut**
Sekce TCP konfiguračního souboru klienta	KeepAlive

Další informace o konfiguraci produktu IBM WebSphere MQ MQI client naleznete v tématu Konfigurace klienta pomocí konfiguračního souboru.

Související úlohy

Trasování aplikací produktu IBM WebSphere MQ classes for Java

Sekce Java Standard Environment Trace

Ke konfiguraci trasovacího prostředku produktu IBM WebSphere MQ classes for Java můžete použít sekci Nastavení trasování standardního prostředí produktu Java .

com.ibm.msg.client.commonservices.trace.outputName = `traceOutputName`

`traceOutputName` je adresář a název souboru, do kterého je odeslán výstup trasování.

Výchozí název trasovacího souboru závisí na verzi produktu IBM WebSphere MQ classes for Java , kterou aplikace používá:

- Pro IBM WebSphere MQ classes for Java for Version 7.5.0, Fix Pack 8 nebo dřívější je *traceOutputNázev* standardně nastaven na soubor s názvem `mqjms_%PID%.trc` v aktuálním pracovním adresáři.
- V 7.5.0.9 Z produktu IBM WebSphere MQ classes for Java z Version 7.5.0, Fix Pack 9se *traceOutputName* standardně nastaví na soubor s názvem `mqjava_%PID%.trc` v aktuálním pracovním adresáři.

kde *%PID%* je ID aktuálního procesu. Je-li ID procesu nedostupné, vygeneruje se náhodné číslo a připojí se jako předpona k písmenu `f`. Chcete-li zahrnout ID procesu do souboru, který zadáte, použijte řetězec `%PID%` .

Uvedete-li alternativní adresář, musí existovat a musíte mít oprávnění k zápisu do tohoto adresáře. Nemáte-li oprávnění k zápisu, bude výstup trasování zapsán do `System.err`.

com.ibm.msg.client.commonservices.trace.include = `includeList`

`includeList` je seznam balíků a tříd, které jsou trasovány, nebo speciální hodnoty `ALL` nebo `NONE`.

Oddělte názvy balíků nebo tříd středníkem (`;`). Výchozí hodnota parametru **`includeList`** je `ALL`a trasuje všechny balíky a třídy v produktu IBM WebSphere MQ classes for Java.

Poznámka: Můžete zahrnout balík, ale pak vyloučit podbalíky tohoto balíku. Pokud například zahrnete balík `a.b` a vyloučíte balík `a.b.x`, trasování zahrnuje vše v `a.b.y` a `a.b.z`, ale ne `a.b.x` nebo `a.b.x.1`.

com.ibm.msg.client.commonservices.trace.exclude = `excludeList`

`excludeList` je seznam balíků a tříd, které nejsou trasovány, nebo speciální hodnoty `ALL` nebo `NONE`.

Oddělte názvy balíků nebo tříd středníkem (`;`). Výchozí hodnota parametru **`excludeList`** je `NONE`, a proto vylučuje žádné balíky a třídy v produktu IBM WebSphere MQ classes for Java , které jsou trasovány.

Poznámka: Balík můžete vyloučit, ale pak zahrnout podbalíky tohoto balíku. Například, pokud vyloučíte balík `a.b` a zahrnete balík `a.b.x`, trasování zahrnuje vše v `a.b.x` a `a.b.x.1`, ale ne `a.b.y` nebo `a.b.z`.

Zahrne se jakýkoli balík nebo třída, která je uvedená, na stejné úrovni, jak je zahrnuta i vyloučená.

com.ibm.msg.client.commonservices.trace.maxBytes = `maxArrayBytes`

`maxArrayBytes` je maximální počet bajtů, které jsou trasovány z libovolných bajtových polí.

Je-li hodnota **`maxArrayBytes`** nastavena na kladné celé číslo, omezuje počet bajtů v bajtovém poli, které jsou zapsány do trasovacího souboru. Ořízne bajtové pole po zápisu *maxArrayBytes*

ven. Nastavení **maxArrayBytes** snižuje velikost výsledného trasovacího souboru a snižuje účinek trasování na výkon aplikace.

Hodnota `0` pro tuto vlastnost znamená, že do souboru trasování se neodešle žádný obsah žádného bajtového pole.

Předvolená hodnota je `-1`, čímž se odstraní libovolný limit počtu bajtů v bajtovém poli, které jsou odeslány do trasovacího souboru.

com.ibm.msg.client.commonservices.trace.limit = maxTraceBytes
maxTraceBytes je maximální počet bajtů, které jsou zapsány do výstupního trasovacího souboru.

Produkt **maxTraceBytes** pracuje s produktem **traceCycles**. Pokud se počet bajtů trasování nachází v blízkosti limitu, soubor se zavře a spustí se nový výstupní soubor trasování.

Hodnota `0` znamená, že výstupní soubor trasování má nulovou délku. Výchozí hodnota je `-1`, což znamená, že množství dat, která mají být zapsána do výstupního souboru trasování, je neomezeno.

com.ibm.msg.client.commonservices.trace.count = traceCycles
traceCycles je počet výstupních souborů trasování, které se mají procházet.

Pokud aktuální trasovací výstupní soubor dosáhne limitu, který je specifikován parametrem **maxTraceBytes**, soubor se zavře. Další výstup trasování se zapisuje do dalšího výstupního souboru trasování v pořadí. Každý trasovací výstupní soubor je rozlišen číselnou příponou, která je připojena k názvu souboru. Aktuální nebo poslední výstupní soubor trasování má příponu `.trc.0`, další nejnovější trasovací výstupní soubor končí znakem `.trc.1`a tak dále. Starší trasovací soubory postupují podle stejného vzoru číslování až do limitu.

Standardní hodnota **traceCycles** je 1. Je-li **traceCycles** 1, dosáhne-li aktuální výstupní soubor trasování jeho maximální velikost, soubor se uzavře a odstraní. Je spuštěn nový výstupní soubor trasování se stejným názvem. Proto v daném okamžiku existuje pouze jeden výstupní soubor trasování.

com.ibm.msg.client.commonservices.trace.parameter = traceParameters
Produkt **traceParameters** řídí, zda jsou parametry metod a návratové hodnoty zahrnuty do trasování.

Výchozí hodnota parametru **traceParameters** je TRUE. Je-li parametr **traceParameters** nastaven na hodnotu FALSE, trasují se pouze podpisy metody.

com.ibm.msg.client.commonservices.trace.compress = compressedTrace
Nastavte **compressedTrace** na TRUE , chcete-li komprimovat výstup trasování.

Výchozí hodnota **compressedTrace** je FALSE.

Je-li parametr **compressedTrace** nastaven na hodnotu TRUE, bude výstup trasování komprimován. Výchozí název výstupního souboru trasování má příponu `.trz`. Je-li komprese nastavena na FALSE, výchozí hodnota, má soubor příponu `.trc` , aby indikoval, že je nekomprimovaný. Je-li však zadán název souboru pro výstup trasování v souboru **traceOutputName** , použije se místo něj název a žádná přípona se nepoužije na soubor.

Komprimovaný výstup trasování je menší než nekomprimovaný. Protože je zde méně vstupů/výstupů, lze jej zapsat rychleji než nekomprimované trasování. Komprimované trasování má menší vliv na výkon IBM WebSphere MQ classes for Java než nekomprimované trasování.

Je-li nastavena hodnota **maxTraceBytes** a **traceCycles** , vytvoří se více komprimovaných trasovacích souborů místo několika prostých textových souborů.

Pokud se IBM WebSphere MQ classes for Java ukončuje nekontrolovaným způsobem, komprimovaný trasovací soubor nemusí být platný. Z tohoto důvodu musí být komprese trasování použita pouze tehdy, je-li IBM WebSphere MQ classes for Java ukončen řízeným způsobem. Kompresi trasování používejte pouze v případě, že problémy, které jsou předmětem vyšetřování, nezpůsobují neočekávané zastavení prostředí JVM. Nepoužívejte kompresi trasování, když diagnostikujete problémy, které mohou mít za následek `System.Halt()` vypnutí nebo nestandardní, nekontrolované ukončení prostředí JVM.

com.ibm.msg.client.commonservices.trace.level = `traceLevel`

`traceLevel` uvádí úroveň filtrování pro trasování. Definovaná úroveň trasování je následující:

Tabulka 87. Co je trasováno pro každou úroveň trasování	
Hodnota	**Co je trasováno**
0	Trasování je vypnuto
1	Výjimky
3	Výjimky Varování
6	Výjimky Varování Informační body trasování
8	Výjimky Varování Informační body trasování Vstup metody a ukončení
9	Výjimky Varování Informační body trasování Vstup metody a ukončení Data, která jsou odeslána mezi produktem IBM WebSphere MQ classes for Java a správcem front.

Poznámka: Vždy použijte hodnotu 9 , pokud není jinak určena podporou produktu IBM .

IBM WebSphere MQ classes for Java a nástroje pro správu softwaru

Nástroje pro správu softwaru, jako např. Apache Maven, lze použít s produktem IBM WebSphere MQ classes for Java.

Řada velkých rozvojových organizací používá tyto nástroje k centrální správě úložišť knihoven jiných dodavatelů.

IBM WebSphere MQ classes for Java se skládá z určitého počtu souborů JAR. Když vyvíjíte aplikace v jazyce Java pomocí tohoto rozhraní API, je na počítači, na kterém je vyvíjována aplikace, vyžadována instalace buď serveru IBM WebSphere MQ , klienta IBM WebSphere MQ , nebo klienta IBM WebSphere MQ SupportPac .

Chcete-li použít nástroj pro správu softwaru a přidat soubory JAR, které tvoří IBM WebSphere MQ classes for Java do centrálně spravovaného úložiště, musí být dodrženy následující body:

- Úložiště nebo kontejner musí být k dispozici pouze pro vývojáře v rámci vaší organizace. Jakékoli rozdělení mimo organizaci není povoleno.
- Úložiště musí obsahovat úplnou a konzistentní sadu souborů JAR z jedné verze produktu IBM WebSphere MQ nebo z opravné sady Fix Pack.
- Jste zodpovědní za aktualizaci úložiště pomocí jakékoli údržby poskytované společností IBM Support.

Pro produkt IBM WebSphere MQ Version 7.5musí být do úložiště nainstalovány následující soubory JAR:

- com.ibm.mq.commonservices.jar
- com.ibm.mq.jar
- com.ibm.mq.jmqi.jar
- com.ibm.mq.pcf.jar
- com.ibm.mq.headers.jar
- connector.jar

Postup po instalaci pro aplikace produktu IBM WebSphere MQ

Po instalaci produktu IBM WebSphere MQmůžete nakonfigurovat svou instalaci tak, abyste spouštěli vlastní aplikace.

Nezapomeňte zkontrolovat soubor README produktu IBM WebSphere MQ pro pozdější nebo více specifických informací pro vaše prostředí.

Před pokusem o spuštění tříd produktu IBM WebSphere MQ pro aplikaci Java v režimu vazeb se ujistěte, že jste nakonfigurovali produkt IBM WebSphere MQ , jak je popsáno v části Konfigurace .

Konfigurace správce front tak, aby přijímal klientská připojení z tříd produktu WebSphere MQ pro prostředí Java

Chcete-li nakonfigurovat správce front tak, aby přijímal příchozí požadavky na připojení od klientů, definujte a povolte použití kanálu připojení k serveru a spusťte program modulu listener.

Podrobnosti viz “Příprava a spuštění ukázkových programů” na stránce 104.

Spuštění tříd WebSphere MQ pro aplikace v jazyce Java pod produktem Java Security Manager

WebSphere MQ Classes for Java může být spuštěn s povoleným produktem Java Security Manager. Chcete-li úspěšně spustit aplikace s povoleným produktem Security Manager, musíte nakonfigurovat prostředí JVM (Java Virtual Machine) s vhodným souborem definic zásad.

Nejjednodušším způsobem, jak to provést, je změnit soubor zásad dodaný spolu s vaším prostředím JRE. Na většině systémů je tento soubor uložen v cestě `lib/security/java.policy`, relativně k adresáři JRE. Soubory zásad můžete upravovat pomocí svého preferovaného editoru nebo pomocí programu `policytool` dodaného s vaším prostředím JRE.

Musíte udělit oprávnění k souboru com.ibm.mq.jmqi.jar tak, aby mohl:

- Vytvoření soketů (v režimu klienta)
- Načíst nativní knihovnu (v režimu vazeb)
- Číst různé vlastnosti z prostředí

Systémová vlastnost os.name musí být k dispozici pro třídy WebSphere MQ pro prostředí Java při spuštění pod produktem Java Security Manager.

Zde je uveden příklad položky souboru zásad, který umožňuje úspěšné spuštění tříd produktu WebSphere MQ pro jazyk Java pod výchozím správcem zabezpečení:

```
grant codeBase "file:MQ_INSTALLATION_PATH/java/lib/com.ibm.mq.jmqi.jar" {
  //Required
  permission java.util.PropertyPermission "user.name","read";
  permission java.util.PropertyPermission "os.name","read";
  //Required if mqclient.ini/mqs.ini configuration files are used
  permission java.io.FilePermission "/var/mqm/mqclient.ini","read";
  permission java.io.FilePermission "/var/mqm/mqs.ini","read";
  //For the client transport type.
  permission java.net.SocketPermission "*","connect";
  //For the bindings transport type.
  permission java.lang.RuntimePermission "loadLibrary.*";
  //For applications that use CCDT tables (access to the CCDT
AMQCLCHL.TAB)
  permission java.io.FilePermission
"/var/mqm/qmgrs/QMGR/@ipcc/AMQCLCHL.TAB","read";
  //For applications that use User Exits
```

```
  permission java.io.FilePermission "/var/mqm/exits/*","read";
  permission java.lang.RuntimePermission "createClassLoader";
  //Required for the z/OS platform
  permission java.util.PropertyPermission
"com.ibm.vm.bitmode","read";
};
grant codeBase
"file:MQ_INSTALLATION_PATH/java/lib/com.ibm.mq.commonservices.jar" {
  permission java.util.PropertyPermission "user.dir","read";
  permission java.util.PropertyPermission "line.separator","read";
  //tracing permissions
  permission java.util.PropertyPermission "com.ibm.mq.commonservices", "read";
  permission java.util.logging.LoggingPermission "control";
  //For access to the trace properties file.
  permission java.io.FilePermission "/tmp/trace.properties", "read";
  //For access to the trace output files.
  permission java.io.FilePermission "/tmp/*", "read,write";
};
```

Notes:

- Produkt *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.
- Tento příklad souboru zásad umožňuje, aby třídy WebSphere MQ pro prostředí Java pracovaly správně pod správcem zabezpečení, ale přesto může být nutné, aby váš vlastní kód fungoval správně, než budete moci aplikace pracovat.
- Chcete-li povolit třídy WebSphere MQ pro Javu přístup k souborům archivu Java (JAR) aplikace, přidejte následující oprávnění k prvnímu příkazu `grant` :

  ```
  permission java.io.FilePermission "/path_to_your_app/-", "read";
  ```

- Chcete-li použít tyto příkazy `grant` v konfiguračním souboru zásad, může být třeba upravit názvy cest v závislosti na tom, kam jste nainstalovali třídy WebSphere MQ pro prostředí Java a kde ukládáte vaše aplikace.
- Ukázkový kód dodávaný s produktem WebSphere MQ Classes for Java nebyl specificky povolen pro použití se správcem zabezpečení; testy IVT však běží s tímto souborem zásad a výchozím správcem zabezpečení je.

Ověření instalace tříd produktu IBM WebSphere MQ pro produkt Java

Program pro ověření instalace, MQIVP, je dodáván s třídami IBM WebSphere MQ pro produkt Java. Tento program můžete použít k testování všech režimů připojení tříd produktu IBM WebSphere MQ pro produkt Java.

Program vás vyzve k zadání řady voleb a jiných dat, abyste určili, který režim připojení chcete ověřit.
K ověření instalace použijte následující proceduru:

1. Chcete-li spustit program v režimu klienta, nakonfigurujte správce front tak, jak je popsáno v tématu “Příprava a spuštění ukázkových programů” na stránce 104. Fronta, která má být použita, je SYSTEM.DEFAULT.LOCAL.QUEUE.
2. Chcete-li spustit program v režimu klienta, přečtěte si také téma “Použití tříd produktu WebSphere MQ pro prostředí Java” na stránce 631.

   Proveďte zbývající kroky tohoto postupu na systému, na kterém chcete spustit program.
3. Ujistěte se, že jste aktualizovali proměnnou prostředí CLASSPATH podle pokynů v části “Proměnné prostředí relevantní pro třídy produktu WebSphere MQ pro prostředí Java” na stránce 635.
4. Změňte adresář na `MQ_INSTALLATION_PATH/mqm/VRM/java/samples/wmqjava`, kde MQ_INSTALLATION_PATH je cesta k vaší instalaci produktu IBM WebSphere MQ a VRM je číslo verze, vydání a modifikace produktu. Pak na příkazový řádek zadejte:

   ```
   java -Djava.library.path=library_path MQIVP
   ```

kde *cesta_knihovny* je cesta ke třídám produktu IBM WebSphere MQ pro knihovny produktu Java (viz Třídy WebSphere MQ pro knihovny Java).

Na výzvu označenou (1):

- Chcete-li použít připojení TCP/IP, zadejte název hostitele serveru IBM WebSphere MQ .
- Chcete-li použít nativní připojení (režim vazeb), ponechte pole prázdné (nezadávejte název).

Program se pokouší:

- 1. Připojte se ke správci front
- 2. Otevřete frontu SYSTEM.DEFAULT.LOCAL.QUEUE, vložte zprávu do fronty, získat zprávu z fronty a poté zavřít frontu
- 3. Odpojení od správce front
- 4. Vrátit zprávu, pokud jsou operace úspěšné

Zde je uveden příklad výzev a odpovědí, které můžete vidět. Skutečné výzvy k zadání a vaše odpovědi závisí na vaší síti IBM WebSphere MQ .

```
Please enter the IP address of the MQ server              : ipaddress(1)
Please enter the port to connect to                       : (1414)(2)
Please enter the server connection channel name            : channelname(2)
Please enter the queue manager name                       : qmname
Success: Connected to queue manager.
Success: Opened SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Put a message to SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Got a message from SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Closed SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Disconnected from queue manager


Tests complete -
SUCCESS: This MQ Transport is functioning correctly.
Press Enter to continue ...
```

Poznámka:

1. Vyberete-li připojení k serveru, nezobrazí se výzvy k zadání (2).

Řešení problémů s produktem IBM WebSphere MQ

Na začátku spusťte program pro ověření instalace. Je možné, že budete muset použít trasovací prostředek.

Pokud se program nedokončí úspěšně, spusťte program pro ověření instalace a postupujte podle pokynů uvedených v diagnostických zprávách. Tento program je popsán v tématu “Ověření instalace tříd produktu IBM WebSphere MQ pro produkt Java” na stránce 644.

Pokud problémy přetrvávají a vy potřebujete kontaktovat servisní tým IBM , můžete být požádáni o zapnutí trasovacího prostředku. Proveďte to tak, jak je uvedeno v následujícím příkladu.

Chcete-li trasovat program `MQIVP` :

- Vytvořte soubor vlastností *com.ibm.mq.commonservices* (viz Použití com.ibm.mq.commonservices .
- Zadejte následující příkaz:

```
java -Dcom.ibm.mq.commonservices=commonservices_properties_file java
-Djava.library.path=library_path MQIVP -trace
```

kde:

- *commonservices_properties_file* je cesta (včetně názvu souboru) k souboru vlastností *com.ibm.mq.commonservices* .
- *cesta_knihovny* je cesta k třídám produktu WebSphere MQ pro knihovny Java (viz Třídy WebSphere MQ pro knihovny Java).

Další informace o tom, jak používat trasování, najdete v tématu Trasování aplikací produktu IBM WebSphere MQ classes for Java.

Úvod pro programátory

Tato kolekce témat obsahuje obecné informace pro programátory.

Další podrobné informace o psaní programů najdete v tématu “Zápis tříd produktu WebSphere MQ pro aplikace Java” na stránce 646.

Třídy WebSphere MQ pro rozhraní Java

Procedurální programovací rozhraní produktu WebSphere MQ používá příkazy, které pracují s objekty. Programovací rozhraní Java používá objekty, se kterými se můžete chovat voláním metod.

Procedurální programovací rozhraní produktu WebSphere MQ je založeno na příkazových slovech, jako jsou tyto:

```
MQBACK, MQBEGIN, MQCLOSE, MQCONN, MQDISC,
MQGET, MQINQ, MQOPEN, MQPUT, MQSET, MQSUB
```

Tato příkazová slova slouží jako parametr k obsluze objektu WebSphere MQ , na kterém mají být provozovány. Vzhledem k tomu, že Java je objektově orientovaná, změní se toto kolo v programovacím rozhraní Java. Váš program se skládá ze sady objektů produktu WebSphere MQ , které se při volání metod na těchto objektech chovají.

Když použijete procedurální rozhraní, odpojíte se od správce front pomocí volání MQDISC (Hconn, CompCode, Reason), kde *Hconn* je manipulátor pro správce front.

V rozhraní Java je správce front reprezentován objektem třídy MQQueueManager. Odpojení od správce front zavoláte voláním metody disconnect () na dané třídě.

```
// declare an object of type queue manager
MQQueueManager queueManager=new MQQueueManager();
...
// do something...
...
// disconnect from the queue manager
queueManager.disconnect();
```

Zápis tříd produktu WebSphere MQ pro aplikace Java

Tato kolekce témat obsahuje informace, které vám pomohou při psaní aplikací Java pro interakci se systémy WebSphere MQ .

Chcete-li používat třídy WebSphere MQ pro jazyk Java pro přístup k frontám produktu WebSphere MQ , budete psát aplikace v jazyku Java, které obsahují volání, do nichž jsou zprávy vloženy a získávat zprávy z front produktu WebSphere MQ . Podrobnosti o jednotlivých třídách naleznete v tématu Třídy WebSphere MQ pro jazyk Java.

Poznámka: Automatické opětovné připojení klienta není podporováno třídami produktu WebSphere MQ pro prostředí Java.

Režimy připojení WebSphere MQ pro režimy připojení Java

Způsob, jakým program pro třídy WebSphere MQ for Java obsahuje některé závislosti na režimech připojení, které chcete použít.

Pokud používáte připojení klienta, existuje řada rozdílů oproti IBM WebSphere MQ MQI client , ale je koncepčně podobná. Používáte-li režim vazeb, můžete použít vazby se zkrácenou cestou a můžete zadat příkaz MQBEGIN. Režim, který má být použit, určujete nastavením proměnných ve třídě MQEnvironment.

třídy připojení klienta WebSphere MQ pro připojení klienta Java

Když se třídy WebSphere MQ pro jazyk Java používají jako klient, je to jako IBM WebSphere MQ MQI client, ale má mnoho rozdílů.

Pokud programujete *Třídy WebSphere MQ pro jazyk Java* pro použití jako klienta, uvědomte si následující rozdíly:

- Podporuje pouze TCP/IP.
- Při spuštění se nečte žádné proměnné prostředí produktu WebSphere MQ .
- Informace, které budou uloženy v definici kanálu a v proměnných prostředí, mohou být uloženy ve třídě s názvem Prostředí. Alternativně lze tyto informace předat jako parametry při vytvoření připojení.
- Chybové stavy a výjimky jsou zapsány do protokolu uvedeného ve třídě `MQException` . Výchozí místo určení chyby je konzola Java.
- Pouze následující atributy v konfiguračním souboru klienta WebSphere MQ jsou relevantní pro třídy produktu WebSphere MQ pro jazyk Java. Určíte-li jiné atributy, jsou neúčinné.

sekce	**Atribut**
ClientExitSekce Cesta ke konfiguračnímu souboru klienta	ExitsDefaultPath
ClientExitSekce Cesta ke konfiguračnímu souboru klienta	ExitsDefaultPath64
ClientExitSekce Cesta ke konfiguračnímu souboru klienta	Cesta ke třídě JavaExits
stanzaMessageBuffer konfiguračního souboru klienta	MaximumSize
stanzaMessageBuffer konfiguračního souboru klienta	PurgeTime
stanzaMessageBuffer konfiguračního souboru klienta	UpdatePercentage
Sekce TCP konfiguračního souboru klienta	ClntRcvBufSize
Sekce TCP konfiguračního souboru klienta	ClntSndBufSize
Sekce TCP konfiguračního souboru klienta	Časový limit připojení
Sekce TCP konfiguračního souboru klienta	KeepAlive

- Při připojení ke správci front, který vyžaduje převod znakových dat, je nyní klient jazyka Java V7 schopen provést převod, pokud to správce front není schopen provést. Prostředí JVM klienta musí podporovat převod mezi CCSID klienta a ID správce front.
- Třídy WebSphere MQ pro jazyk Java automatické opětovné připojování klientů nepodporují.

Při použití v klientském režimu produkt *WebSphere MQ classes for Java* nepodporuje volání MQBEGIN.

Další informace o podporovaných prostředích viz “Volby připojení pro třídy WebSphere MQ pro prostředí Java” na stránce 632 .

Režim vazeb WebSphere MQ pro režim vazeb Java

Režim vazeb tříd produktu WebSphere MQ pro prostředí Java se liší od režimu klienta třemi hlavními způsoby.

Při použití v režimu vazeb třídy WebSphere MQ pro jazyk Java používají rozhraní JNI (Java Native Interface) k volání přímo do existujícího rozhraní API správce front, a nikoli prostřednictvím komunikace prostřednictvím sítě.

Aplikace, které používají třídy WebSphere MQ pro prostředí Java v režimu vazeb, se při výchozím nastavení připojují ke správci front pomocí volby *ConnectOption*, MQCNO_STANDARD_BINDINGS.

Třídy WebSphere MQ pro prostředí Java podporují následující *ConnectOptions*:

- VAZBA MQCNO_FASTPATH_BINDING
- VAZBA MQCNO_STANDARD_BINDING
- CQCNO_SHARED_BINDING
- VAZBA MQCNO_ISOLATED_BINDING

Další informace o *ConnectOptions*viz “Připojení ke správci front pomocí volání MQCONNX” na stránce 201.

Režim vazeb podporuje volání MQBEGIN za účelem zahájení globálních pracovních jednotek koordinovaných správcem front na všech platformách s výjimkou produktu WebSphere MQ pro produkt IBM i a WebSphere MQ pro systém z/OS.

Většina parametrů poskytnutých třídou `MQEnvironment` není důležitá pro režim vazeb a je ignorována.

Další informace o podporovaných prostředích viz “Volby připojení pro třídy WebSphere MQ pro prostředí Java” na stránce 632 .

Definování, které třídy produktu WebSphere MQ pro připojení Java se mají použít

Typ připojení, který má být použit, je určen nastavením proměnných ve třídě MQEnvironment.

Jsou použity dvě proměnné:

MQEnvironment.properties
Typ připojení je určen hodnotou přidruženou k názvu klíče `CMQC.TRANSPORT_PROPERTY`. Možné hodnoty jsou následující:

CMQC.TRANSPORT_MQSERIES_BINDINGS
Připojit se v režimu vazeb

CMQC.TRANSPORT_MQSERIES_CLIENT
Připojit v režimu klienta

CMQC.TRANSPORT_MQSERIES
Režim připojení je určen hodnotou vlastnosti *hostname* .

MQEnvironment.hostname
Nastavte hodnotu této proměnné následujícím způsobem:

- U klientských připojení nastavte hodnotu této proměnné na název hostitele serveru IBM WebSphere MQ , ke kterému se chcete připojit.
- Pro režim vazeb nenastavujte tuto proměnnou, nebo ji nastavte na hodnotu null.

Operace pro správce front

Tato kolekce témat popisuje způsob připojení a odpojení od správce front s použitím tříd WebSphere MQ pro prostředí Java.

Nastavení prostředí produktu WebSphere MQ pro třídy WebSphere MQ pro prostředí Java

Aby se aplikace mohla připojit ke správci front v režimu klienta, musí aplikace určit název kanálu, název hostitele a číslo portu.

Poznámka: Informace v tomto tématu jsou relevantní pouze v případě, že se vaše aplikace připojuje ke správci front v režimu klienta. Pokud se připojuje v režimu vazeb, je *nevhodná* . Viz “Režimy připojení pro třídy WebSphere MQ pro JMS” na stránce 747.

Můžete zadat název kanálu, název hostitele a číslo portu jedním ze dvou způsobů: buď jako pole ve třídě MQEnvironment, nebo jako vlastnosti objektu MQQueueManager .

Nastavíte-li pole ve třídě MQEnvironment, použijí se na celou aplikaci, kromě případů, kdy jsou přepsány transformační tabulkou vlastností. Chcete-li určit název kanálu a název hostitele v prostředí MQEnvironment, použijte následující kód:

```
MQEnvironment.hostname = "host.domain.com";
MQEnvironment.channel  = "java.client.channel";
```

To je ekvivalentní nastavení proměnné prostředí **MQSERVER** :

```
"java.client.channel/TCP/host.domain.com".
```

Při výchozím nastavení se klienti Java pokusí připojit k modulu listener produktu WebSphere MQ na portu 1414. Chcete-li určit jiný port, použijte následující kód:

```
MQEnvironment.port = nnnn;
```

kde nnnn je požadované číslo portu.

Předáte-li vlastnosti objektu správce front při jeho vytvoření, použijí se pouze pro daného správce front. Vytvořte položky v objektu Hashtable s klíči **hostname**, **channel**a volitelně s hodnotami **port**a s příslušnými hodnotami. Chcete-li použít výchozí port, 1414, můžete vynechat položku **port** . Vytvořte objekt MQQueueManager pomocí konstruktoru, který přijímá hašovací tabulku vlastností.

Identifikace připojení ke správci front nastavením názvu aplikace

Aplikace může nastavit název, který identifikuje její připojení ke správci front. Tento název aplikace je zobrazen pomocí příkazu **DISPLAY CONN MQSC/PCF** (kde pole se nazývá **APPLTAG**) nebo v zobrazení WebSphere MQ Explorer **Připojení aplikace** (kde se pole nazývá **App name**).

Názvy aplikací jsou omezeny na 28 znaků a delší názvy jsou zkráceny tak, aby se vešly. Není-li zadán název aplikace, je poskytnuta výchozí hodnota. Výchozí název je založen na vyvolání (hlavní) třídě, ale pokud tyto informace nejsou k dispozici, použije se text `WebSphere MQ Client for Java` .

Je-li použit název vyvolávající třídy, je v případě potřeby upraven tak, aby se odstranily úvodní názvy balíků. Je-li například třída vyvolání `com.example.MainApp`, použije se úplný název, ale pokud je třída vyvolání `com.example.dictionaryAndThesaurus.multilingual.mainApp`, použije se název `multilingual.mainApp` , protože se jedná o nejdelší kombinaci názvu třídy a názvu balíku nejvíce vpravo, který se vejde do dostupné délky.

Je-li název třídy delší než 28 znaků, je oříznut na vhodný. Například `com.example.mainApplicationForSecondTestCase` se stane `mainApplicationForSecondTest`.

Poznámka: Správci front spuštění na platformách z/OS nepodporují nastavení názvů aplikací.

Chcete-li nastavit název aplikace ve třídě MQEnvironment, přidejte název do hašovací tabulky MQEnvironment.properties s klíčem **MQConstants.APPNAME_PROPERTY**pomocí následujícího kódu:

```
MQEnvironment.properties.put(MQConstants.APPNAME_PROPERTY, "my_application_name");
```

Chcete-li nastavit název aplikace v hašovací tabulce vlastností, která se předává konstruktoru MQQueueManager , přidejte název do hašovací tabulky vlastností s klíčem **MQConstants.APPNAME_PROPERTY**.

Potlačení vlastností uvedených v konfiguračním souboru klienta WebSphere MQ

Konfigurační soubor klienta WebSphere MQ může také určovat vlastnosti, které se používají ke konfiguraci tříd WebSphere MQ pro prostředí Java. Vlastnosti zadané v konfiguračním souboru klienta WebSphere MQ MQI se však používají pouze v případě, že se aplikace připojuje ke správci front v režimu klienta.

V případě potřeby můžete přepsat libovolný atribut v konfiguračním souboru produktu WebSphere MQ některým z následujících způsobů. Volby jsou zobrazeny v pořadí podle priority.

- Nastavte systémovou vlastnost Java pro vlastnost konfigurace.
- Nastavte tuto vlastnost v mapě MQEnvironment.properties .
- V systému Java5 a novějších verzích nastavte systémovou proměnnou prostředí.

Pouze následující atributy v konfiguračním souboru klienta WebSphere MQ jsou relevantní pro třídy produktu WebSphere MQ pro prostředí Java. Pokud uvedete nebo přepisují jiné atributy, nemá žádný efekt.

sekce	**Atribut**
ClientExitSekce Cesta ke konfiguračnímu souboru klienta	ExitsDefaultPath
ClientExitSekce Cesta ke konfiguračnímu souboru klienta	ExitsDefaultPath64
ClientExitSekce Cesta ke konfiguračnímu souboru klienta	Cesta ke třídě JavaExits
stanzaMessageBuffer konfiguračního souboru klienta	MaximumSize
stanzaMessageBuffer konfiguračního souboru klienta	PurgeTime
stanzaMessageBuffer konfiguračního souboru klienta	UpdatePercentage
Sekce TCP konfiguračního souboru klienta	ClntRcvBufSize
Sekce TCP konfiguračního souboru klienta	ClntSndBufSize
Sekce TCP konfiguračního souboru klienta	Časový limit připojení
Sekce TCP konfiguračního souboru klienta	KeepAlive

Připojení ke správci front ve třídách produktu WebSphere MQ pro prostředí Java

Připojte se ke správci front vytvořením nové instance třídy MQQueueManager . Odpojení od správce front voláním metody disconnect ().

Nyní jste připraveni připojit se ke správci front vytvořením nové instance třídy MQQueueManager :

```
MQQueueManager queueManager = new MQQueueManager("qMgrName");
```

Chcete-li se odpojit od správce front, volejte metodu disconnect () ve správci front:

```
queueManager.disconnect();
```

Pokud zavoláte metodu odpojení, všechny otevřené fronty a procesy, ke kterým jste v daném správci front přistoupil, jsou zavřeny. Je však dobrým programovacím postupem, abyste tyto prostředky při jejich používání explicitně uzavřeli. Chcete-li to provést, použijte metodu close () na příslušných objektech.

Metody commit () a backout () ve správci front jsou ekvivalentní volání MQCMIT a MQBACK, které se používají s procedurálním rozhraním.

Použití tabulky definic kanálů klienta s IBM WebSphere MQ classes for Java

Klientská aplikace IBM WebSphere MQ classes for Java může používat definice kanálů připojení klienta uložené v tabulce definic kanálů klienta (CCDT).

Jako alternativa k vytvoření definice kanálu připojení klienta nastavením určitých polí a vlastností prostředí ve třídě `MQEnvironment` nebo jejich předáním do `MQQueueManager` v transformační tabulce vlastností může klientská aplikace IBM WebSphere MQ classes for Java používat definice kanálů připojení klienta, které jsou uloženy v tabulce definic kanálů klienta. Tyto definice jsou vytvořeny příkazy skriptuIBM WebSphere MQ Script (MQSC) nebo příkazy PCF (IBM WebSphere MQ Programmable Command Format) nebo pomocí nástroje IBM WebSphere MQ Explorer.

Když aplikace vytvoří objekt `MQQueueManager` , klient IBM WebSphere MQ classes for Java prohledá tabulku definic kanálů klienta pro vhodnou definici kanálu připojení klienta a použije definici kanálu ke spuštění kanálu MQI. Další informace o tabulkách definic kanálů klienta a o tom, jak je vytvořit, najdete v tématu Tabulka definic kanálů klienta.

Chcete-li použít tabulku definic kanálů klienta, musí aplikace nejprve vytvořit objekt URL. Objekt URL zapouzdřuje adresu URL, která identifikuje název a umístění souboru obsahujícího tabulku definic kanálů klienta a určuje, jak lze k souboru přistupovat.

Pokud například soubor `ccdt1.tab` obsahuje tabulku definic kanálů klienta a je uložen ve stejném systému, v němž je aplikace spuštěna, může aplikace vytvořit objekt adresy URL následujícím způsobem:

```
java.net.URL chanTab1 = new URL("file:///home/admdata/ccdt1.tab");
```

Jako další příklad předpokládejme, že soubor `ccdt2.tab` obsahuje tabulku definic kanálů klienta a je uložen v systému, který se liší od tabulky, na které je aplikace spuštěna. Pokud k souboru lze přistupovat pomocí protokolu FTP, může aplikace vytvořit objekt adresy URL následujícím způsobem:

```
java.net.URL chanTab2 = new URL("ftp://ftp.server/admdata/ccdt2.tab");
```

Poté, co aplikace vytvoří objekt URL, může aplikace vytvořit objekt `MQQueueManager` pomocí jednoho z konstruktorů, který vezme objekt URL jako parametr. Příklad:

```
MQQueueManager mars = new MQQueueManager("MARS", chanTab2);
```

Tento příkaz způsobí, že klient IBM WebSphere MQ classes for Java má přístup k tabulce definic kanálů klienta identifikovanému objektem adresy URL chanTab2, prohledá tabulku pro vhodnou definici kanálu připojení klienta a poté použije definici kanálu ke spuštění kanálu MQI pro správce front s názvem MARS.

Všimněte si následujících bodů, které se použijí, pokud aplikace používá tabulku definic kanálů klienta:

- Když aplikace vytvoří objekt `MQQueueManager` pomocí konstruktoru, který přijímá objekt URL jako parametr, nesmí být ve třídě `MQEnvironment` nastaven žádný název kanálu, a to buď jako pole, nebo jako vlastnost prostředí. Je-li nastaven název kanálu, klient IBM WebSphere MQ classes for Java vygeneruje `MQException`. Vlastnost pole nebo prostředí určující název kanálu je považován za nastavený, pokud jeho hodnota obsahuje jinou hodnotu než null, prázdný řetězec nebo řetězec obsahující všechny prázdné znaky.
- Parametr **queueManagerName** v konstruktoru `MQQueueManager` může mít jednu z následujících hodnot:
  - Název správce front
  - Hvězdička (*) následovaná názvem skupiny správců front
  - Hvězdička (*)
  - Null, prázdný řetězec, nebo řetězec obsahující všechny prázdné znaky

  Jedná se o tytéž hodnoty, které lze použít pro parametr **QMgrName** v rámci volání MQCONN vydaného aplikací klienta, která používá rozhraní MQI (Message Queue Interface). Další informace o významu těchto hodnot najdete v tématu “Přehled rozhraní fronty zpráv” na stránce 187.

  Pokud vaše aplikace používá fondy připojení, prohlédněte si téma “Řízení výchozího fondu připojení ve třídách WebSphere MQ pro prostředí Java” na stránce 671.

- Pokud klient IBM WebSphere MQ classes for Java najde vhodnou definici kanálu pro připojení klienta v tabulce definic kanálů klienta, použije pouze informace extrahované z této definice kanálu ke spuštění kanálu MQI. Všechny pole související s kanálem nebo vlastnosti prostředí, které aplikace může mít nastaveny ve třídě `MQEnvironment` , se budou ignorovat.

  Zejména si všimněte následujících bodů, používáte-li zabezpečení SSL (Secure Sockets Layer):

  – Kanál MQI používá zabezpečení SSL pouze v případě, že definice kanálu extrahovaná z definiční tabulky kanálu klienta určuje název CipherSpec podporovaný klientem IBM WebSphere MQ classes for Java .
  – Tabulka definic kanálů klienta také obsahuje informace o umístění serverů LDAP (Lightweight Directory Access Protocol), které uchovávají seznamy zrušených certifikátů (CRL). Klient IBM WebSphere MQ classes for Java používá pouze tyto informace pro přístup k serverům LDAP, které obsahují seznamy odvolaných certifikátů.
  – Definiční tabulka kanálu klienta může také obsahovat umístění odpovídacího modulu OCSP (Online Certificate Status Protocol). IBM WebSphere MQ classes for Java nemůže použít informace OCSP v souboru tabulky definic kanálů klienta. Nicméně můžete OCSP nakonfigurovat podle popisu uvedeného v kapitole Používání protokolu certifikátů online.

  Další informace o použití zabezpečení SSL s tabulkou definic kanálů klienta naleznete v tématu Určení, zda kanál MQI používá zabezpečení SSL.

  Všimněte si také následujících bodů, pokud používáte uživatelské procedury kanálu:

  – Kanál MQI používá uživatelské procedury kanálu a přidružená uživatelská data určená prostřednictvím definice kanálu extrahované z tabulky definic kanálů klienta v preferovaném kanálu a data zadaná pomocí jiných metod.
  – Definice kanálu extrahovaná z tabulky definic kanálů klienta může určovat uživatelské procedury kanálu, které jsou zapsány v produktu Java, C nebo C + +. Další informace o tom, jak zapisovat uživatelskou proceduru kanálu v produktu Java, naleznete v tématu “Vytvoření uživatelské procedury kanálu v produktu WebSphere MQ classes for Java” na stránce 664. Další informace o tom, jak zapisovat uživatelskou proceduru kanálu v jiných jazycích, najdete v tématu “Použití uživatelských procedur kanálu nepsaných v jazyce Java s třídami WebSphere MQ pro prostředí Java” na stránce 668.

Určení rozsahu portů pro připojení klienta IBM WebSphere MQ classes for Java

Můžete uvést port, nebo rozsah portů, které může aplikace svázat dvěma způsoby.

Pokusí-li se aplikace IBM WebSphere MQ classes for Java o připojení ke správci front produktu IBM WebSphere MQ v režimu klienta, brána firewall může povolit pouze ta připojení, která pocházejí ze zadaných portů nebo rozsahu portů. V této situaci můžete uvést port nebo rozsah portů, ke kterým se aplikace může vázat. Port (y) můžete zadat následujícími způsoby:

- Můžete nastavit pole localAddressSetting ve třídě MQEnvironment. Příklad:

```
MQEnvironment.localAddressSetting = "192.0.2.0(2000,3000)";
```

- Můžete nastavit vlastnost prostředí CMQC.LOCAL_ADDRESS_PROPERTY. Příklad:

```
(MQEnvironment.properties).put(CMQC.LOCAL_ADDRESS_PROPERTY,
                               "192.0.2.0(2000,3000)");
```

- Když můžete sestavit objekt MQQueueManager , můžete předat hašovací tabulku vlastností obsahující LOCAL_ADDRESS_PROPERTY hodnotou "192.0.2.0(2000,3000)".

V každém z těchto příkladů, když se aplikace později připojí ke správci front, se aplikace připojí k lokální adrese IP a číslu portu v rozsahu 192.0.2.0(2000) na 192.0.2.0(3000).

V systému s více než jedním síťovým rozhraním můžete také použít pole Nastavení localAddressnebo vlastnost prostředí CMQC.LOCAL_ADDRESS_PROPERTY, chcete-li určit, které síťové rozhraní musí být použito pro připojení.

Pokud omezíte rozsah portů, může dojít k chybám připojení. Dojde-li k chybě, dojde k výjimce MQException obsahující kód příčiny IBM WebSphere MQ MQRC_Q_MGR_NOT_AVAILABLE a následující zpráva:

```
Socket connection attempt refused due to LOCAL_ADDRESS_PROPERTY restrictions
```

Může se vyskytnout chyba, pokud jsou použity všechny porty v uvedeném rozsahu, nebo pokud zadaná adresa IP, název hostitele nebo číslo portu nejsou platné (například záporné číslo portu).

Přístup k frontám, tématům a procesům ve třídách produktu WebSphere MQ pro prostředí Java

Chcete-li přistupovat k frontám, tématům a procesům, použijte metody třídy MQQueueManager . MQOD (struktura deskriptoru objektu) je sbalena do parametrů těchto metod.

Fronty

Chcete-li otevřít frontu, můžete použít metodu accessQueue třídy MQQueueManager . Například u správce front s názvem queueManagerpoužijte následující kód:

```
MQQueue queue = queueManager.accessQueue("qName",CMQC.MQOO_OUTPUT);
```

Metoda accessQueue vrací nový objekt třídy MQQueue.

Až skončíte s používáním fronty, použijte metodu close () k její zavření, jako v následujícím příkladu:

```
queue.close();
```

Frontu lze vytvořit také pomocí konstruktoru MQQueue. Parametry jsou přesně stejné jako u metody accessQueue spolu s parametrem správce front. Příklad:

```
MQQueue queue = new MQQueue(queueManager,
                            "qName",
                            CMQC.MQOO_OUTPUT,
                            "qMgrName",
                            "dynamicQName",
                            "altUserID");
```

Při vytváření front můžete zadat řadu voleb. Podrobnosti o těchto údajích naleznete v dokumentu Class.com.ibm.mq.MQQueue. Při vytváření objektu fronty tímto způsobem lze zapisovat do vlastních podtříd fronty MQQueue.

Témata

Podobně je možné otevřít téma pomocí metody accessTopic třídy MQQueueManager . Například ve správci front s názvem queueManagerpoužijte k vytvoření odběratele a vydavatele následující kód:

```
MQTopic subscriber =
      queueManager.accessTopic("TOPICSTRING","TOPICNAME",
      CMQC.MQTOPIC_OPEN_AS_SUBSCRIPTION, CMQC.MQSO_CREATE);
```

```
MQTopic publisher =
      queueManager.accessTopic("TOPICSTRING","TOPICNAME",
      CMQC.MQTOPIC_OPEN_AS_PUBLICATION, CMQC.MQOO_OUTPUT);
```

Po dokončení používání daného tématu ji zavřete pomocí metody close ().

Rovněž můžete vytvořit téma pomocí konstruktoru MQTopic. Parametry jsou přesně stejné jako u metody accessTopic spolu s přidáním parametru správce front. Příklad:

```
MQTopic subscriber = new
      MQTopic(queueManager,"TOPICSTRING","TOPICNAME",
      CMQC.MQTOPIC_OPEN_AS_SUBSCRIPTION, CMQC.MQSO_CREATE);
```

Při vytváření témat můžete zadat řadu voleb. Podrobné informace o těchto tématech naleznete v tématu Třída com.ibm.mq.MQTopic . Při vytváření objektu tématu v tomto způsobu lze psát vlastní podtřídy MQTopic.

Téma musí být otevřeno buď pro publikování, nebo pro odběr. Třída MQQueueManager má osm metod accessTopic a třída Topic má osm konstruktorů. V každém případě čtyři z nich mají parametr **destination** a čtyři mají parametr **subscriptionName** (včetně dvou, které mají obě). Ty lze použít pouze k otevření tématu pro odběry. Tyto dvě zbývající metody mají parametr **openAs** a lze je otevřít buď pro publikování, nebo pro odběr, v závislosti na hodnotě parametru **openAs** .

Chcete-li vytvořit téma jako trvalý odběratel, použijte buď metodu accessTopic třídy MQQueueManager , nebo konstruktor MQTopic, který přijímá název odběru, a v obou případech nastavte položku CMQC.MQSO_DURABLE .

Procesy

Chcete-li přistupovat k procesu, použijte metodu accessProcess objektu MQQueueManager. Například ve správci front s názvem queueManagervytvořte objekt MQProcess pomocí následujícího kódu:

```
MQProcess process =
      queueManager.accessProcess("PROCESSNAME",
      CMQC.MQOO_FAIL_IF_QUIESCING);
```

Chcete-li přistupovat k procesu, použijte metodu accessProcess objektu MQQueueManager.

Metoda accessProcess vrací nový objekt třídy MQProcess.

Když jste dokončili použití objektu procesu, použijte metodu close () k zavření, jako v následujícím příkladu:

```
process.close();
```

Proces můžete také vytvořit pomocí konstruktoru MQProcess. Parametry jsou přesně stejné jako u metody accessProcess spolu s přidáním parametru správce front. Příklad:

```
 MQProcess process =
      new MQProcess(queueManager,"PROCESSNAME",
      CMQC.MQOO_FAIL_IF_QUIESCING);
```

Konstrukce objektu procesu tímto způsobem umožňuje zápis do vlastních podtříd procesu MQProcess.

Zpracování zpráv ve třídách produktu WebSphere MQ pro prostředí Java

Zprávy jsou reprezentovány třídou MQMessage. Pomocí metod třídy MQDestination, která má podtřídy MQQueue a MQTopic, jste umístili a získali zprávy pomocí metod třídy MQDestination.

Vložení zpráv do front nebo témat pomocí metody put () třídy MQDestination. Zprávy získáte z front nebo témat pomocí metody get () třídy MQDestination. Na rozdíl od procedurálního rozhraní, kde příkazy MQPUT a MQGET vložit a získat pole bajtů, uvede programovací jazyk Java a získává instance třídy MQMessage. Třída MQMessage zapouzdřuje vyrovnávací paměť dat, která obsahuje skutečná data zprávy, spolu se všemi parametry MQMD (deskriptor zprávy) a vlastnostmi zpráv, které tuto zprávu popisují.

Chcete-li vytvořit novou zprávu, vytvořte novou instanci třídy MQMessage a pomocí metod writeXXX umístěte data do vyrovnávací paměti zpráv.

Když se vytvoří nová instance zprávy, všechny parametry MQMD se automaticky nastaví na jejich výchozí hodnoty, jak je definováno v Počáteční hodnoty a deklarace jazyka pro MQMD . Metoda put () objektu

MQDestination také používá instanci třídy voleb MQPutMessagejako parametr. Tato třída představuje strukturu MQPMO. Následující příklad vytvoří zprávu a vloží ji do fronty:

```
// Build a new message containing my age followed by my name
MQMessage myMessage = new MQMessage();
myMessage.writeInt(25);

String name = "Charlie Jordan";
myMessage.writeInt(name.length());
myMessage.writeBytes(name);

// Use the default put message options...
MQPutMessageOptions pmo = new MQPutMessageOptions();

// put the message!
queue.put(myMessage,pmo);
```

Metoda get () MQDestination vrací novou instanci MQMessage, která představuje zprávu, která byla právě převzata z fronty. Jako parametr má také instanci třídy voleb MQGetMessage. Tato třída představuje strukturu MQGMO.

Maximální velikost zprávy není třeba zadávat, protože metoda get () automaticky upravuje velikost vnitřní vyrovnávací paměti tak, aby se vešla do příchozí zprávy. K přístupu k datům ve vrácené zprávě použijte metody readXXX třídy MQMessage.

Následující příklad uvádí, jak získat zprávu z fronty:

```
// Get a message from the queue
MQMessage theMessage    = new MQMessage();
MQGetMessageOptions gmo = new MQGetMessageOptions();
queue.get(theMessage,gmo);  // has default values

// Extract the message data
int age = theMessage.readInt();
int strLen = theMessage.readInt();
byte[] strData = new byte[strLen];
theMessage.readFully(strData,0,strLen);
String name = new String(strData,0);
```

Formát čísla, který metody čtení a zápisu používá, můžete změnit nastavením proměnné člena *encoding* .

Můžete změnit znakovou sadu, která má být použita pro čtení a zápis řetězců, a to nastavením proměnné člena *characterSet* .

Další informace viz “Třída MQMessage” na stránce 1036.

Poznámka: Metoda writeUTF() MQMessage automaticky kóduje délku řetězce stejně jako bajty Unicode, které obsahuje. Když bude vaše zpráva čtena jiným programem Java (pomocí readUTF()), je to nejjednodušší způsob, jak odeslat informace o řetězci.

Zlepšení výkonu přechodných zpráv ve třídách WebSphere MQ pro prostředí Java

Chcete-li zlepšit výkon při procházení zpráv nebo přijímání přechodných zpráv z aplikace klienta, můžete použít *čtení napřed*. Klientské aplikace používající příkaz MQGET nebo asynchronní spotřeba budou těžit ze zlepšení výkonu při procházení zpráv nebo při přijímání přechodných zpráv.

Obecné informace o prostředku pro dopředné čtení najdete v tématu související téma.

V třídách WebSphere MQ pro jazyk Java můžete použít vlastnosti CMQC.MQSO_READ_AHEAD a CMQC.MQSO_NO_READ_AHEAD objektu MQQueue nebo MQTopic k určení toho, zda mají spotřebitelé zpráv a prohlížečové prohlížeče povoleno používat dopředné čtení u daného objektu.

Asynchronně vkládání zpráv pomocí tříd WebSphere MQ pro jazyk Java

Chcete-li asynchronně vložit zprávu, nastavte MQPMO_ASYNC_RESPONSE.

Do front nebo témat vkládejte zprávy do front nebo témat pomocí metody put () třídy MQDestination. Chcete-li asynchronně vložit zprávu, tj. umožnění dokončení operace bez čekání na odezvu ze správce

front, můžete nastavit MQPMO_ASYNC_RESPONSE v poli voleb volby MQPutMessage. Chcete-li určit úspěch nebo selhání asynchronních operací vložení, použijte volání funkce MQQueueManager.getAsync.

Publikování/odběr ve třídách produktu WebSphere MQ pro prostředí Java

Ve třídách produktu WebSphere MQ pro jazyk Java je téma reprezentováno třídou MQTopic a pomocí metod MQTopic.put() pro něj publikujete.

Obecné informace o produktu WebSphere MQ publish/subscribe naleznete v části Úvod do systému zpráv publikování/odběru WebSphere MQ .

Zpracování záhlaví zpráv produktu WebSphere MQ s třídami produktu WebSphere MQ pro prostředí Java

Poskytují se třídy Java představující různé typy záhlaví zprávy. Poskytují se také dvě pomocné třídy.

Objekty záhlaví jsou popsány rozhraním MQHeader, které poskytuje všestranně určené metody pro přístup k polím záhlaví a pro čtení a zápis obsahu zpráv. Každý typ záhlaví má svou vlastní třídu, která implementuje rozhraní MQHeader a přidává metody getter a setter pro jednotlivá pole. Například typ záhlaví MQRFH2 je reprezentován třídou MQRFH2 ; typ záhlaví MQDLH třídou MQDLH atd. Třídy záhlaví provádějí veškerá nezbytná konverze dat automaticky a mohou číst nebo zapisovat data v libovolném uvedeném numerickém kódování nebo znakové sadě (CCSID).

Dvě nápovědné třídy, MQHeaderIterator a MQHeaderListpomáhají při čtení a dekódování (analyzování) obsahu záhlaví ve zprávách:

- Třída MQHeaderIterator funguje podobně jako java.util.Iterator. Pokud je ve zprávě více záhlaví, metoda next () vrací hodnotu true a metoda nextHeader() nebo next () vrací další objekt záhlaví.
- MQHeaderList funguje podobně jako java.util.List. Podobně jako MQHeaderIteratoranalyzuje obsah záhlaví, ale také vám umožňuje hledat konkrétní záhlaví, přidávat nová záhlaví, odebírat existující záhlaví, aktualizovat pole záhlaví a poté zapsat obsah záhlaví zpět do zprávy. Případně můžete vytvořit prázdný objekt MQHeaderList, poté jej naplnit instancemi záhlaví a zapsat jej do zprávy jednou nebo opakovaně.

Třídy MQHeaderIterator a MQHeaderList používají informace v souboru MQHeaderRegistry , aby věděli, které třídy záhlaví produktu WebSphere MQ jsou přidruženy ke konkrétním typům a formátům zpráv. Objekt MQHeaderRegistry je konfigurován se znalostmi všech aktuálních formátů a typů záhlaví produktu WebSphere MQ a jejich implementačních tříd a také můžete registrovat své vlastní typy záhlaví.

Podpora je poskytována pro následující běžně používané záhlaví Websphere MQ

- MQRFH-Pravidla a záhlaví formátování
- MQRFH2 -Like MQRFH, který se používá k předávání zpráv do zprostředkovatele zpráv náležícího do produktu WebSphere Message Broker nebo od zprostředkovatele zpráv. Používá se také k uchování vlastností zprávy
- MQCIH-most CICS
- MQDLH-Záhlaví nedoručených zpráv
- MQIIH-záhlaví informací IMS
- MQRMH-záhlaví zprávy odkazu
- MQSAPH-záhlaví SAP
- MQWIH-Záhlaví informací o práci
- MQXQH-záhlaví přenosové fronty
- MQDH-záhlaví distribuce
- MQEPH-zapouzdřené záhlaví PCF

Také můžete definovat třídy reprezentující vaše vlastní záhlaví.

Chcete-li použít MQHeaderIterator k získání záhlaví RFH2 , nastavte buď MQGMO_PROPERTIES_FORCE_MQRFH2 ve volbách GetMessage, nebo nastavte vlastnost fronty PROPCTL na hodnotu FORCE.

Tisk všech záhlaví ve zprávě pomocí tříd produktu WebSphere MQ pro prostředí Java

V tomto příkladu instance třídy MQHeaderIterator analyzuje záhlaví ve frontě zpráv MQMessage, která byla přijata z fronty. Objekty MQHeader vrácené z metody nextHeader() zobrazují jejich strukturu a obsah při vyvolání metody toString .

```
import com.ibm.mq.MQMessage;
import com.ibm.mq.headers.MQHeader;
import com.ibm.mq.headers.MQHeaderIterator;
...
MQMessage message = ... // Message received from a queue.
MQHeaderIterator it = new MQHeaderIterator (message);

while (it.hasNext ())
{
    MQHeader header = it.nextHeader ();

    System.out.println ("Header type " + header.type () + ": " + header);
}
```

Přeskakování záhlaví ve zprávě pomocí tříd WebSphere MQ pro prostředí Java

V tomto příkladu funkce skipHeaders() funkce MQHeaderIterator umístí zprávu na čtení zprávy okamžitě za poslední záhlaví.

```
import com.ibm.mq.MQMessage;
import com.ibm.mq.headers.MQHeaderIterator;
...
MQMessage message = ... // Message received from a queue.
MQHeaderIterator it = new MQHeaderIterator (message);

it.skipHeaders ();
```

Vyhledání kódu příčiny ve zprávě o mrtvém znaku pomocí tříd produktu WebSphere MQ pro prostředí Java

V tomto příkladu metoda čtení naplní objekt MQDLH čtením ze zprávy. Po operaci čtení je kurzor pro čtení zprávy umístěn okamžitě za obsah záhlaví MQDLH.

Zprávy ve frontě nedoručených zpráv správce front mají předponu záhlaví zablokovaných dopisů (MQDLH). Chcete-li se rozhodnout, jak tyto zprávy zpracovat-například k určení, zda mají být pokusy o opakované pokusy nebo jejich vyřazení, musí se podívat na kód příčiny obsažený v souboru MQDLH.

```
import com.ibm.mq.MQMessage;
import com.ibm.mq.headers.MQDLH;
...
MQMessage message = ... // Message received from the dead-letter queue.
MQDLH dlh = new MQDLH ();

dlh.read (message);

System.out.println ("Reason: " + dlh.getReason ());
```

Všechny třídy záhlaví také poskytují konstruktor convenience pro inicializaci přímo ze zprávy v jednom kroku. Takže kód v tomto příkladu by mohl být zjednodušen následujícím způsobem:

```
import com.ibm.mq.MQMessage;
import com.ibm.mq.headers.MQDLH;
...
MQMessage message = ... // Message received from the dead-letter queue.
MQDLH dlh = new MQDLH (message);

System.out.println ("Reason: " + dlh.getReason ());
```

Čtení a odebrání MQDLH ze zprávy s dead-letter pomocí tříd produktu WebSphere MQ pro prostředí Java

V tomto příkladu se MQDLH používá k odstranění záhlaví ze zprávy s dead-letter.

Aplikace pro zpracování smrtelného dopisu bude obvykle znovu odeslat zprávy, které byly zamítnuty, pokud jejich kód příčiny indikuje přechodnou chybu. Před opětovným odesláním zprávy musí být odebráno záhlaví MQDLH.

Tento příklad provádí následující kroky (viz komentáře v ukázkovém kódu):

1. Příkaz MQHeaderList přečte celou zprávu a každé záhlaví, které se v této zprávě objeví, se stane položkou v seznamu.
2. Zprávy nedoručených zpráv obsahují záhlaví MQDLH jako jejich první záhlaví, takže lze tuto položku najít v první položce seznamu záhlaví. Objekt MQDLH již byl naplněn daty ze zprávy, když je sestaven objekt MQHeaderList , takže není třeba volat jeho metodu čtení.
3. Kód příčiny je extrahován pomocí metody getReason() poskytnuté třídou MQDLH.
4. Kód příčiny byl zkontrolován a označuje, že je vhodné znovu odeslat zprávu. MQDLH se odebere pomocí metody remove () MQHeaderList .
5. Funkce MQHeaderList zapisuje svůj zbývající obsah do nového objektu zprávy. Nová zpráva nyní obsahuje vše v původní zprávě s výjimkou MQDLH a může být zapsána do fronty. Argument **true** konstruktoru a metodě write označuje, že tělo zprávy má být zadrženo v rámci MQHeaderLista že je znovu zapsáno.
6. Pole formátu v deskriptoru zprávy nové zprávy nyní obsahuje hodnotu, která byla dříve uvedena v poli formátu MQDLH. Data zprávy se shodují s číselným kódováním a sadou CCSID v deskriptoru zpráv.

```
import com.ibm.mq.MQMessage;
import com.ibm.mq.headers.MQDLH;
import com.ibm.mq.headers.MQHeaderList;
...
MQMessage message = ... // Message received from the dead-letter queue.
MQHeaderList list = new MQHeaderList (message, true); // Step 1.
MQDLH dlh = (MQDLH) list.get (0); // Step 2.
int reason = dlh.getReason (); // Step 3.
...
list.remove (dlh); // Step 4.

MQMessage newMessage = new MQMessage ();

list.write (newMessage, true); // Step 5.
newMessage.format = list.getFormat (); // Step 6.
```

Tisk obsahu zprávy pomocí tříd produktu WebSphere MQ pro prostředí Java

Tento příklad používá MQHeaderList k vytištění obsahu zprávy včetně jeho záhlaví.

Výstup obsahuje zobrazení veškerého obsahu hlavičky a také těla zprávy. Třída MQHeaderList dekóduje všechna záhlaví v jednom kroku, zatímco kroky MQHeaderIterator postupně procházejí procesem pod kontrolou aplikace. Tuto techniku můžete použít k poskytnutí jednoduchého nástroje ladění při zápisu aplikací produktu Websphere MQ .

```
import com.ibm.mq.MQMessage;
import com.ibm.mq.headers.MQHeaderList;
...
MQMessage message = ... // Message received from a queue.

System.out.println (new MQHeaderList (message, true));
```

Tento příklad také vytiskne pole deskriptoru zpráv pomocí třídy MQMD. Metoda copyFrom() třídy com.ibm.mq.headers.MQMD naplní objekt záhlaví daty z polí deskriptoru zpráv MQMessage, nikoli čtením těla zprávy.

```
import com.ibm.mq.MQMessage;
import com.ibm.mq.headers.MQMD;
import com.ibm.mq.headers.MQHeaderList;
...
MQMessage message = ...
```

```
MQMD md = new MQMD ();
...
md.copyFrom (message);
System.out.println (md + "\n" + new MQHeaderList (message, true));
```

Vyhledání specifického typu záhlaví ve zprávě pomocí tříd produktu WebSphere MQ pro prostředí Java

Tento příklad používá metodu indexOf(String) objektu MQHeaderList k nalezení záhlaví MQRFH2 ve zprávě, je-li k dispozici.

```
import com.ibm.mq.MQMessage;
import com.ibm.mq.headers.MQHeaderList;
import com.ibm.mq.headers.MQRFH2;
...
MQMessage message = ...
MQHeaderList list = new MQHeaderList (message);
int index = list.indexOf ("MQRFH2");

if (index >= 0)
{
    MQRFH2 rfh = (MQRFH2) list.get (index);
    ...
}
```

Analýza záhlaví MQRFH2 pomocí tříd WebSphere MQ pro prostředí Java

Tento příklad ukazuje, jak získat přístup ke známé hodnotě pole v pojmenované složce, pomocí třídy MQRFH2 .

Třída MQRFH2 poskytuje řadu způsobů, jak přistupovat nejen k polím v pevné části struktury, ale také k obsahu složek kódovaným XML, které jsou přenášeny v poli Data NameValue. Tento příklad ukazuje, jak získat přístup ke známé hodnotě pole ve jmenované složce-v této instanci, pole Rto ve složce jms, která představuje název fronty odpovědí ve zprávě MQ JMS.

```
MQRFH2 rfh = ...

String value = rfh.getStringFieldValue ("jms", "Rto");
```

Chcete-li zjistit obsah MQRFH2 (na rozdíl od požadavku přímo na specifická pole), můžete použít metodu getFolders k vrácení seznamu MQRFH2.Element, který představuje strukturu složky, která může obsahovat pole a další složky. Nastavení pole nebo složky na hodnotu null ji odebere ze struktury MQRFH2. Když manipulujete s obsahem datové složky NameValuetímto způsobem, pole StrucLength se automaticky aktualizuje odpovídajícím způsobem.

Čtení a zápis toků bajtů jiných než objekty MQMessage pomocí tříd produktu WebSphere MQ pro jazyk Java

Tyto příklady používají třídy záhlaví k analýze a manipulaci s obsahem záhlaví WebSphere MQ v případě, že zdrojem dat není objekt MQMessage.

Třídy záhlaví lze použít k analýze a manipulaci s obsahem záhlaví produktu WebSphere MQ i v případě, že je zdrojem dat jiný objekt než objekt MQMessage. Rozhraní MQHeader, které je implementováno každou třídou záhlaví, poskytuje metody `int read (java.io.DataInput message, int encoding, int characterSet)` a `int write (java.io.DataOutput message, int encoding, int characterSet)`. Třída com.ibm.mq.MQMessage implementuje rozhraní java.io.DataInput a java.io.DataOutput . To znamená, že můžete použít dvě metody MQHeader ke čtení a zápisu obsahu MQMessage, přičemž bude potlačeno kódování a CCSID určený v deskriptoru zpráv. To je užitečné pro zprávy, které obsahují řetězec záhlaví v různých kódování.

Můžete také získat objekty DataInput a DataOutput z jiných toků dat, například souborů nebo proudů soketů nebo bajtových polí přenesených ve zprávách platformy JMS. Třídy java.io.DataInputStream implementují třídy DataInput a třídy java.io.DataOutputStream implementují DataOutput. Tento příklad čte obsah záhlaví WebSphere MQ z bajtového pole:

```
import java.io.*;
import com.ibm.mq.headers.*;
```

```
...
byte [] bytes = ...
DataInput in = new DataInputStream (new ByteArrayInputStream (bytes));
MQHeaderIterator it = new MQHeaderIterator (in, CMQC.MQENC_NATIVE,
  CMQC.MQCCSI_DEFAULT);
```

Řádek, který spouští MQHeaderIterator , může být nahrazen pomocí

```
MQDLH dlh = new MQDLH (in, CMQC.MQENC_NATIVE, CMQC.MQCCSI_DEFAULT);
// or any other header type
```

Tento příklad zapisuje na bajtové pole pomocí proudu DataOutput:

```
MQHeader header = ... // Could be any header type
ByteArrayOutputStream out = new ByteArrayOutputStream ();

header.write (new DataOutputStream (out), CMQC.MQENC_NATIVE, CMQC.MQCCSI_DEFAULT);
byte [] bytes = out.toByteArray ();
```

Když pracujete s proudy tímto způsobem, buďte opatrní, abyste použili správné hodnoty pro kódování a argumenty characterSet . Při čtení záhlaví zadejte kódování a CCSID, se kterým byl původně zapsán bajtový obsah. Při zápisu záhlaví zadejte kódování a CCSID, které chcete vytvořit. Převod dat se provádí automaticky třídami záhlaví.

Vytvoření tříd pro nové typy záhlaví pomocí tříd WebSphere MQ pro prostředí Java

Můžete vytvořit třídy jazyka Java pro typy záhlaví, které nejsou dodávány s třídami produktu WebSphere MQ pro prostředí Java.

Chcete-li přidat třídu Java představující nový typ záhlaví, který můžete použít stejným způsobem jako všechny třídy záhlaví dodané s třídami produktu WebSphere MQ pro prostředí Java, vytvoříte třídu, která implementuje rozhraní MQHeader. Nejjednodušším přístupem je rozšířit třídu com.ibm.mq.headers.impl.Header . Tento příklad vytvoří plně funkční třídu reprezentující strukturu záhlaví MQTM. Nemusíte přidávat jednotlivé metody getter a setter pro každé pole, ale je to užitečné pohodlí pro uživatele třídy záhlaví. Generické metody getValue a setValue , které přijmou řetězec pro název pole, budou pracovat pro všechna pole definovaná v typu záhlaví. Zděděné metody čtení, zápisu a velikosti umožní instancím nového typu záhlaví, které mají být čteny a zapsány, a vypočítá velikost záhlaví správně na základě definice pole. Definice typu se vytvoří právě jednou, ale vytvoří se mnoho instancí této třídy záhlaví. Chcete-li zpřístupnit novou definici záhlaví pro dekódování pomocí tříd MQHeaderIterator nebo MQHeaderList , zaregistrujete ji pomocí MQHeaderRegistry. Všimněte si však, že třída záhlaví MQTM je ve skutečnosti již v tomto balíku poskytnuta a registrována ve výchozím registru.

```
import com.ibm.mq.headers.impl.Header;
import com.ibm.mq.headers.impl.HeaderField;
import com.ibm.mq.headers.CMQC;

public class MQTM extends Header {
    final static HeaderType TYPE = new HeaderType ("MQTM");
    final static HeaderField StrucId = TYPE.addMQChar ("StrucId", CMQC.MQTM_STRUC_ID);
    final static HeaderField Version = TYPE.addMQLong ("Version", CMQC.MQTM_VERSION_1);
    final static HeaderField QName = TYPE.addMQChar ("QName", CMQC.MQ_Q_NAME_LENGTH);
    final static HeaderField ProcessName = TYPE.addMQChar ("ProcessName",
      CMQC.MQ_PROCESS_NAME_LENGTH);
    final static HeaderField TriggerData = TYPE.addMQChar ("TriggerData",
      CMQC.MQ_TRIGGER_DATA_LENGTH);
    final static HeaderField ApplType = TYPE.addMQLong ("ApplType");
    final static HeaderField ApplId = TYPE.addMQChar ("ApplId", 256);
    final static HeaderField EnvData = TYPE.addMQChar ("EnvData", 128);
    final static HeaderField UserData = TYPE.addMQChar ("UserData", 128);

    protected MQTM (HeaderType type){
        super (type);
    }
    public String getStrucId () {
        return getStringValue (StrucId);
    }
    public int getVersion () {
        return getIntValue (Version);
    }
    public String getQName () {
        return getStringValue (QName);
```

```
    }
    public void setQName (String value) {
        setStringValue (QName, value);
    }
    // ...Add convenience getters and setters for remaining fields in the same way.
}
```

Zpracování zpráv PCF s třídami produktu WebSphere MQ pro prostředí Java

Třídy Java jsou poskytovány pro vytváření a analýzu zpráv strukturovaných PCF a pro usnadnění odesílání požadavků PCF a shromažďování odpovědí PCF.

Třídy PCFMessage & MQCFGR představují pole struktur parametrů PCF. Poskytují pohodlnější metody pro přidávání a načítání parametrů PCF.

Struktury parametrů PCF jsou reprezentovány třídami MQCFH, MQCFIN, MQCFIN64, MQCFST, MQCFBS, MQCFIL, MQCFIL64 MQCFSL a MQCFGR. Tato sdílená základní operační rozhraní:

- Metody pro čtení a zápis obsahu zpráv: read (), write () a size ()
- Metody pro manipulaci s parametry: getValue (), setValue (), getParameter () a další
- Metoda výčtového nástroje.nextParameter (), která analyzuje obsah PCF ve zprávě MQMessage

Parametr filtru PCF se používá v dotazových příkazech k poskytnutí funkce filtru. Zapouzdřeno v následujících třídách:

- MQCFIF-celočíselný filtr
- MQCFSF-filtr řetězců
- MQCFBF-filtr bajtů

Jsou poskytnuty dvě třídy agenta, PCFAgent a PCFMessageAgent pro správu připojení ke správci front, frontu příkazového serveru a přidruženou frontu odpovědí. PCFMessageAgent rozšiřuje PCFAgent a měl by se normálně používat v předvolbě. Třída PCFMessageAgent převádí přijaté MQMessages a předává je zpět volajícímu jako pole PCFMessage. Modul PCFAgent vrací pole zpráv MQMessages, které je třeba před použitím analyzovat.

Zpracování vlastností zpráv ve třídách produktu WebSphere MQ pro prostředí Java

Volání funkcí ke zpracování obslužných rutin zpráv nemají ekvivalent v třídách WebSphere MQ pro prostředí Java. Chcete-li nastavit, vrátit nebo odstranit vlastnosti obslužné rutiny zpráv, použijte metody třídy MQMessage.

Obecné informace o vlastnostech zprávy viz “Názvy vlastností” na stránce 18.

Ve třídách produktu WebSphere MQ pro přístup jazyka Java ke zprávám se používá třída MQMessage. Popisovače zpráv nejsou proto v prostředí Java poskytovány a neexistuje žádný ekvivalent k volání funkcí produktu WebSphere MQ MQCRTMH, MQDLTMH, MQMHBUF a MQBUFMH

Chcete-li nastavit vlastnosti zpracování zpráv v procedurálním rozhraní, použijte volání MQSETMP volání. V třídách produktu WebSphere MQ pro jazyk Java použijte příslušnou metodu třídy MQMessage:

- Vlastnost setBoolean
- Vlastnost setByte
- Vlastnost setBytes
- Vlastnost setShort
- Vlastnost setInt
- setInt2Property
- setInt4Property
- setInt8Property
- Vlastnost setLong

- Vlastnost setFloat
- Vlastnost setDouble
- Vlastnost setString
- Vlastnost setObject

Ty se někdy souhrnně označují jako metody *set*property* .

Chcete-li vrátit hodnotu vlastností manipulátoru zprávy v procedurálním rozhraní, použijte volání MQINQMP volání. V třídách produktu WebSphere MQ pro jazyk Java použijte příslušnou metodu třídy MQMessage:

- Vlastnost getBoolean
- Vlastnost getByte
- Vlastnost getBytes
- Vlastnost getShort
- Vlastnost getInt
- getInt2Property
- getInt4Property
- getInt8Property
- Vlastnost getLong
- Vlastnost getFloat
- Vlastnost getDouble
- Vlastnost getString
- Vlastnost getObject

Tyto metody jsou někdy souhrnně označovány jako metody *get*property* .

Chcete-li odstranit hodnotu vlastností zpracování zpráv v procedurálním rozhraní, použijte příkaz MQDLTMP volání. Ve třídách produktu WebSphere MQ pro jazyk Java použijte metodu deleteProperty třídy MQMessage.

Zpracování chyb ve třídách produktu WebSphere MQ pro prostředí Java

Zpracovávat chyby vyplývající ze tříd produktu WebSphere MQ pro jazyk Java s použitím bloků Java `try` a `catch` .

Metody v rozhraní Java nevracejí kód dokončení a kód příčiny. Místo toho vygenerují výjimku vždy, když kód dokončení a kód příčiny pocházející z volání produktu WebSphere MQ nejsou obě nula. Tím se zjednodušíte logiku programu, takže nemusíte vracet návratové kódy po každém volání do produktu WebSphere MQ. Můžete se rozhodnout, ve kterých bodech ve vašem programu se chcete vypořádat s možností selhání. V těchto bodech můžete obklopovat kód pomocí bloků `try` a `catch` , jako v následujícím příkladu:

```
try {
    myQueue.put(messageA,putMessageOptionsA);
    myQueue.put(messageB,putMessageOptionsB);
}
catch (MQException ex) {
    // This block of code is only executed if one of
    // the two put methods gave rise to a non-zero
    // completion code or reason code.
    System.out.println("An error occurred during the put operation:" +
                           "CC = " + ex.completionCode +
                           "RC = " + ex.reasonCode);
    System.out.println("Cause exception:" + ex.getCause() );
}
```

Kódy příčiny volání příkazu WebSphere MQ ohlášené zpět v prostředí Java pro z/OS jsou dokumentovány v publikaci Kódy příčiny pro z/OS a Kódy příčiny pro všechny ostatní platformy.

Do protokolu se zapisují také výjimky, které jsou vyvoláné, zatímco jsou spuštěny třídy WebSphere MQ pro aplikaci Java. Aplikace však může volat metodu MQException.logExclude(), a zabránit tak protokolování výjimek přidružených k určitému kódu příčiny. Možná to budete chtít provést v situacích, kdy očekáváte mnoho výjimek přidružených ke specifickému kódu příčiny, který má být vyhozen, a nechcete, aby byl protokol naplněn těmito výjimkami. Pokud se například vaše aplikace pokusí o získání zprávy z fronty pokaždé, když se opakuje okolo cyklu, a u většiny z těchto pokusů neočekáváte, že ve frontě není vhodná zpráva, můžete zabránit tomu, aby byly zaprotokolovány výjimky přidružené k kódu příčiny MQRC_NO_MSG_AVAILABLE. Pokud aplikace již dříve zaprotokolovala výjimky přidružené ke specifickému kódu příčiny, může povolit, aby tyto výjimky byly protokolovány znovu voláním metody MQException.logInclude().

Někdy kód příčiny nepředává všechny podrobnosti přidružené k chybě. Pro každou výjimku, která je vygenerována, by aplikace měla zkontrolovat připojenou výjimku. Samotná připojená výjimka může mít jinou propojené výjimky a propojené výjimky tvoří řetězec vedoucí zpět k původnímu základnímu problému. Propojená výjimka je implementována použitím mechanizmu výjimek v řetězové výjimce třídy java.lang.Throwable a aplikace obdrží připojenou výjimku voláním metody Throwable.getCause(). Z výjimky, která je instancí MQException, načte MQException.getCause() základní instanci com.ibm.mq.jmqi.JmqiExceptiona getCause z této výjimky načte základní java.lang.Exception , která způsobila chybu.

Třída MQException standardně automaticky ukládá výjimky do souboru System.err, který je obvykle směrován na konzolu. Chcete-li zastavit výjimky objevující se na konzole, zahrňte do vaší aplikace řádek, který bude nastaven na hodnotu MQException.log= null.

Získání a nastavení hodnot atributu ve třídách produktu WebSphere MQ pro prostředí Java

Metody getXXX() a setXXX() jsou poskytovány pro mnoho obecných atributů. K ostatním lze přistupovat pomocí generických metod zjišťování () a set ().

U mnoha obecných atributů třídy MQManagedObject, MQDestination, MQQueue, MQTopic, MQProcess a MQQueueManager obsahují metody getXXX() a setXXX(). Tyto metody umožňují získat a nastavit hodnoty atributů. Všimněte si, že pro objekty MQDestination, MQQueue a MQTopic pracují metody pouze tehdy, zadáte-li při otevření objektu příslušné parametry dotazu a nastavení.

Pro méně běžné atributy jsou všechny třídy MQQueueManager, MQDestination, MQQueue, MQTopic a MQProcess všechny dědit ze třídy s názvem MQManagedObject. Tato třída definuje dotazová rozhraní () a set ().

Když vytváříte nový objekt správce front pomocí operátoru *nový* , je automaticky otevřen pro zjištění. Použijete-li metodu accessProcess() k přístupu k objektu procesu, je tento objekt automaticky otevřen pro zjišťování. Použijete-li metodu accessQueue() k přístupu k objektu fronty, tento objekt *není* automaticky otevřen pro dotazování nebo nastavení operací. Důvodem je to, že přidání těchto voleb automaticky může způsobit problémy s některými typy vzdálených front. Chcete-li ve frontě použít metody zjišťování, nastavení, getXXXa setXXX , musíte v parametru openOptions metody accessQueue() určit příslušné parametry dotazu a nastavit příznaky. To samé platí pro cílové objekty a objekty témat.

Metody dotazování a nastavení mají tři parametry:

- pole selektorů
- Pole intAttrs
- pole charAttrs

Nepotřebujete parametry SelectorCount, IntAttrCount a CharAttrLength, které se nacházejí v MQINQ, protože délka pole v jazyce Java je vždy známá. Následující příklad uvádí, jak provést dotaz na frontu:

```
// inquire on a queue
final static int MQIA_DEF_PRIORITY = 6;
final static int MQCA_Q_DESC = 2013;
final static int MQ_Q_DESC_LENGTH = 64;

int[] selectors  = new int[2];
```

```
int[] intAttrs   = new int[1];
byte[] charAttrs = new byte[MQ_Q_DESC_LENGTH]

selectors[0] = MQIA_DEF_PRIORITY;
selectors[1] = MQCA_Q_DESC;

queue.inquire(selectors,intAttrs,charAttrs);

System.out.println("Default Priority = " + intAttrs[0]);
System.out.println("Description : " + new String(charAttrs,0));
```

Vícevláknové programy v jazyce Java

Běhové prostředí Java je z podstaty vícevláknové. Třídy WebSphere MQ pro jazyk Java umožňují, aby objekt správce front byl sdílen více podprocesy, ale zajišťuje synchronizaci všech přístupů k cílovému správci front.

Vícevláknové programy se mohou v prostředí Java vyhnout. 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 z podstaty vícevláknové. Proto se inicializace vaší aplikace vyskytuje v jednom podprocesu a kód, který se provádí v odpovědi na stisknutí tlačítka, se provádí v samostatném podprocesu (podproces uživatelského rozhraní).

Při použití klienta WebSphere MQ MQI založeného na jazyce C by tento problém způsobil problém, protože sdílení manipulátorů s více podprocesy má určitá omezení. Třídy WebSphere MQ pro prostředí Java toto omezení umožňují, aby objekt správce front (a jeho přidružená fronta, téma a objekty procesu) byl sdílen více podprocesy.

Implementace tříd produktu WebSphere MQ pro prostředí Java zajišťuje, že pro určité připojení (instance objektuMQQueueManager) bude veškerý přístup k cílovému správci front WebSphere MQ synchronizován. Podproces, který chce vydat volání správci front, je blokován, dokud nejsou dokončena všechna ostatní volání v průběhu tohoto připojení. Pokud vyžadujete souběžný přístup ke stejnému správci front z více podprocesů ve svém programu, vytvořte nový objekt MQQueueManager pro každý podproces, který vyžaduje souběžný přístup. (Toto je ekvivalent k zadání samostatného volání MQCONN pro každé vlákno.)

Poznámka: Instance třídy `com.ibm.mq.MQGetMessageOptions` nesmí být sdíleny mezi podprocesy, které současně vyžadují zprávy. Instance této třídy se aktualizují daty během odpovídající žádosti MQGET, což může mít za následek neočekávané důsledky, pokud více podprocesů souběžně pracuje na stejné instanci objektu.

Použití kanálů kanálů v třídách WebSphere MQ pro prostředí Java

Přehled způsobu použití kanálů v aplikaci s použitím tříd produktu WebSphere MQ pro prostředí Java.

Následující témata popisují způsob zápisu uživatelské procedury kanálu v jazyce Java, jak ji přiřadit a jak do něj předávat data. Pak popisují, jak se používají uživatelské procedury kanálu napsané v C a jak používat sled uživatelských procedur.

Vaše aplikace musí mít nastaveno správné oprávnění zabezpečení pro načtení třídy uživatelských procedur kanálu.

Vytvoření uživatelské procedury kanálu v produktu WebSphere MQ classes for Java

Můžete poskytnout svůj vlastní kanál tak, že definujete třídu Java, která implementuje příslušné rozhraní.

Chcete-li implementovat uživatelskou proceduru, definujte novou třídu Java, která implementuje příslušné rozhraní. Tři výstupní rozhraní jsou definována v balíku com.ibm.mq.exits :

- WMQSendExit
- WMQReceiveExit
- WMQSecurityExit

Poznámka: Uživatelské procedury kanálu jsou podporovány pouze pro připojení klienta; nejsou podporovány pro připojení vazeb. Nemůžete použít uživatelskou proceduru kanálu Java mimo třídy WebSphere MQ pro jazyk Java, například pokud používáte aplikaci klienta napsanou v jazyce C.

Jakékoli šifrování SSL definované pro připojení se provádí *po* vyvolání a byly vyvolány uživatelské procedury pro zabezpečení odeslání a zabezpečení. Podobně se dešifrování provádí *před* a vyvoláno ukončení zabezpečení.

Následující ukázka definuje třídu, která implementuje všechna tři rozhraní:

```
public class MyMQExits implements
WMQSendExit, WMQReceiveExit, WMQSecurityExit {
    // Default constructor
  public MyMQExits(){
  }
    // This method comes from the send exit interface
  public ByteBuffer channelSendExit(
MQCXP channelExitParms,
                                    MQCD channelDefinition,
                                    ByteBuffer agentBuffer)
  {
    // Fill in the body of the send exit here
  }
    // This method comes from the receive exit interface
  public ByteBuffer channelReceiveExit(
MQCXP channelExitParms,
                                       MQCD channelDefinition,
                                       ByteBuffer agentBuffer)
  {
    // Fill in the body of the receive exit here
  }
    // This method comes from the security exit interface
  public ByteBuffer channelSecurityExit(
MQCXP channelExitParms,
                                        MQCD channelDefinition,
                                        ByteBuffer agentBuffer)
  {
    // Fill in the body of the security exit here
  }
}
```

Každé uživatelské proceduře je předán objekt MQCXP a objekt MQCD. Tyto objekty reprezentují struktury MQCXP a MQCD definované v procedurálním rozhraní.

Každá třída ukončení, kterou napíšete, musí mít konstruktor. Může se jednat buď o výchozí konstruktor, nebo o takový, který bude mít řetězcový argument. Pokud je řetězec zapotřebí, budou data uživatele předána do třídy ukončení při jejím vytvoření. Pokud třída uživatelské procedury obsahuje jak výchozí konstruktor, tak konstruktor s jedním argumentem, má konstruktor s jedním argumentem prioritu.

V případě uživatelských procedur pro odesílání a zabezpečení musí návratový kód vracet data, která chcete odeslat na server. V případě uživatelské procedury příjmu musí návratový kód vracet upravená data, která mají být interpretována jako WebSphere MQ .

Nejjednodušším možným výstupním tělem je:

```
{ return agentBuffer; }
```

Nezavírejte správce front v rámci uživatelské procedury kanálu.

Použití existujících tříd uživatelské procedury kanálu

Ve verzích produktu WebSphere MQ starších než 7.0byla tato uživatelská procedura použita pomocí rozhraní MQSendExit, MQReceiveExita MQSecurityExit, jak je uvedeno v následujícím příkladu. Tato metoda zůstává platná, ale nová metoda je upřednostňována pro zlepšení funkčnosti a výkonu.

```
public class MyMQExits implements MQSendExit, MQReceiveExit, MQSecurityExit {
    // Default constructor
  public MyMQExits(){
  }
    // This method comes from the send exit
```

```
  public byte[] sendExit(MQChannelExit channelExitParms,
                         MQChannelDefinition channelDefParms,
                         byte agentBuffer[])
  {
    // Fill in the body of the send exit here
  }
    // This method comes from the receive exit
  public byte[] receiveExit(MQChannelExit channelExitParms,
                            MQChannelDefinition channelDefParms,
                            byte agentBuffer[])
  {
    // Fill in the body of the receive exit here
  }
    // This method comes from the security exit
  public byte[] securityExit(MQChannelExit channelExitParms,
                             MQChannelDefinition channelDefParms,
                             byte agentBuffer[])
  {
    // Fill in the body of the security exit here
  }
}
```

Přiřazení ukončení kanálu v produktu IBM WebSphere MQ classes for Java

Uživatelská procedura kanálu můžete přiřadit pomocí produktu IBM WebSphere MQ classes for Java.

V produktu IBM WebSphere MQ classes for Javaneexistuje žádný přímý ekvivalent k kanálu IBM WebSphere MQ . Uživatelské procedury kanálu jsou přiřazeny k objektu MQQueueManager. Definujete-li například třídu, která implementuje rozhraní WMQSecurityExit , může aplikace použít uživatelskou proceduru zabezpečení jedním ze čtyř způsobů:

- Přiřazením instance třídy do pole MQEnvironment.channelSecurityExit před vytvořením objektu MQQueueManager
- Nastavením pole MQEnvironment.channelSecurityExit na řetězec reprezentující třídu uživatelské procedury zabezpečení před vytvořením objektu MQQueueManager .
- Vytvořením dvojice klíč/hodnota v hašovací tabulce vlastností předané aplikaci MQQueueManager s klíčem CMQC.SECURITY_EXIT_PROPERTY .
- Použití tabulky CCDT (Client Channel Definition table)

Každá uživatelská procedura přiřazená nastavením pole MQEnvironment.channelSecurityExit na řetězec, vytvoření dvojice klíč/hodnota v tabulce vlastností vlastností nebo použití tabulky CCDT, musí být napsána s výchozím konstruktorem. Ukončení přiřazené jako instance třídy nepotřebuje výchozí konstruktor, v závislosti na aplikaci.

Aplikace může podobným způsobem použít odeslání nebo přijetí pro přijetí. Například následující fragment kódu ukazuje, jak používat uživatelské procedury zabezpečení, odeslání a přijetí implementované ve třídě MyMQExits, která byla definována dříve pomocí prostředí MQEnvironment:

```
MyMQExits myexits = new MyMQExits();
MQEnvironment.channelSecurityExit = myexits;
MQEnvironment.channelSendExit = myexits;
MQEnvironment.channelReceiveExit = myexits;
:
MQQueueManager jupiter = new MQQueueManager("JUPITER");
```

Je-li k přiřazení uživatelské procedury kanálu použita více než jedna metoda, bude mít pořadí přednosti následující:

1. Je-li adresa URL tabulky CCDT předána do správce MQQueueManager, obsah kanálu CCDT určuje ukončení kanálu, které má být použito, a všechny definice ukončení v prostředí MQEnvironment nebo hašovací tabulka vlastností jsou ignorovány.
2. Není-li předána žádná adresa URL tabulky CCDT, dojde ke sloučení definic ukončení z prostředí MQEnvironment a hašovací tabulky.
   - Je-li definován stejný typ ukončení v rozhraní MQEnvironment i v tabulce hashtable, použije se definice v tabulce hashtable.

- Jsou-li zadány stejné staré a nové typy ukončení (například pole sendExit , které lze použít pouze pro typ ukončení použité ve verzích produktu IBM WebSphere MQ starších než verze 7.0a channelSend, které lze použít pro všechny uživatelské procedury odeslání), použije se nová uživatelská procedura (channelSendExit) namísto staré uživatelské procedury.

Pokud jste deklarovali uživatelskou proceduru kanálu jako řetězec, musíte produkt IBM WebSphere MQ povolit, aby nalezl ukončovací program kanálu. Můžete tak učinit různými způsoby v závislosti na prostředí, ve kterém je aplikace spuštěna, a na tom, jak jsou uživatelské programy kanálu zabaleny.

- Pro aplikaci spuštěnou na aplikačním serveru musíte uložit soubory do adresáře zobrazeného v produktu Tabulka 88 na stránce 667 nebo zabalené do souborů JAR odkazovaných produktem **exitClasspath**.
- V případě aplikace, která není spuštěna na aplikačním serveru, platí následující pravidla:
  - Pokud jsou vaše třídy ukončení kanálu zabaleny do samostatných souborů JAR, tyto soubory JAR musí být obsaženy v produktu **exitClasspath**.
  - Nejsou-li třídy uživatelské procedury kanálu zabaleny v souborech JAR, lze soubory tříd uložit do adresáře zobrazeného v produktu Tabulka 88 na stránce 667 nebo do libovolného adresáře v cestě ke třídám systému JVM nebo **exitClasspath**.

Vlastnost **exitClasspath** může být zadána čtyřmi způsoby. V zájmu priority jsou tyto způsoby následující:

1. Systémová vlastnost com.ibm.mq.exitClasspath (definovaná na příkazovém řádku pomocí volby `-D`).
2. Stanza exitPath souboru `mqclient.ini`
3. Záznam hašovací tabulky s klíčem CMQC.EXIT_CLASSPATH_PROPERTY
4. Proměnná MQEnvironment **exitClasspath**

Oddělte více cest pomocí znaku java.io.File.pathSeparator .

Tabulka 88. Adresář pro uživatelské programy kanálu

Platforma	**Adresář**
AIX, HP-UX, Linuxa Solaris	`/var/mqm/exits` (programy s 32bitovým ukončovacím programem kanálu) `/var/mqm/exits64` (programy s 64bitovým ukončovacím programem)
Windows	*instalační_dat_adr*\exits
Poznámka: *instalační_dat_adr* je adresář, který jste vybrali pro datové soubory produktu IBM WebSphere MQ během instalace. Standardní adresář je `C:\Program Files\IBM\WebSphere MQ`.	

Předání dat do ukončení kanálu ve třídách produktu WebSphere MQ pro prostředí Java

Můžete předávat data do ukončení kanálu a vracet data z kanálů z kanálů do vaší aplikace.

Parametr agentBuffer

Pro uživatelskou proceduru odeslání zprávy obsahuje parametr *agentBuffer* data, která se mají odeslat. Pro uživatelskou proceduru pro přijetí zprávy nebo ukončení zabezpečení obsahuje parametr *agentBuffer* data, která právě byla přijata. Nepotřebujete mít parametr délky, protože výraz agentBuffer.limit () označuje délku pole.

V případě uživatelských procedur pro odesílání a zabezpečení musí návratový kód vracet data, která chcete odeslat na server. V případě uživatelské procedury příjmu musí návratový kód vracet upravená data, která mají být interpretována jako WebSphere MQ .

Nejjednodušším možným výstupním tělem je:

```
{ return agentBuffer; }
```

Uživatelské procedury kanálu jsou volány s vyrovnávací pamětí, která má záložní pole. Pro nejlepší výkon by měla uživatelská procedura vrátit vyrovnávací paměť s záložním polem.

Data uživatele

Pokud se aplikace připojí ke správci front nastavením volby channelSecurityExit, channelSendExit nebo channelReceiveExit, lze do příslušné třídy uživatelských procedur kanálu předat 32 bajtů uživatelských dat pomocí polí channelSecurityExitUserData, channelSendExitUserData, nebo channelReceiveExitUserData. Tato uživatelská data jsou k dispozici pro třídu uživatelské procedury kanálu, ale je aktualizována při každém zavolání uživatelské procedury. Jakékoli změny provedené v uživatelských datech v uživatelské proceduře kanálu budou proto ztraceny. Chcete-li provádět trvalé změny dat v uživatelské proceduře kanálu, použijte oblast MQCXP exitUser. Data v tomto poli se udržují mezi vyvoláními ukončení.

Pokud aplikace nastaví securityExit, sendExitnebo receiveExit, nelze do těchto tříd uživatelské procedury kanálu předat žádná uživatelská data.

Pokud aplikace používá tabulku definic kanálů klienta (CCDT) k připojení ke správci front, budou veškerá uživatelská data zadaná v definici kanálu připojení klienta předávána třídám uživatelské procedury kanálu při jejich volání. Další informace o použití tabulky definic kanálů klienta viz “Použití tabulky definic kanálů klienta s IBM WebSphere MQ classes for Java” na stránce 650.

Použití uživatelských procedur kanálu nepsaných v jazyce Java s třídami WebSphere MQ pro prostředí Java

Způsob použití ukončovacích programů kanálu napsaných v jazyce C z aplikace Java.

V produktu WebSphere MQ verze 7.0můžete zadat název uživatelského programu kanálu napsaného v jazyce C jako řetězec předaný do polí channelSecurityExit, channelSendExit nebo channelReceivev objektu MQEnvironment nebo vlastnosti Hashtable vlastností. Nelze však použít uživatelskou proceduru kanálu napsanou v jazyce Java v aplikaci napsané v jiném jazyce.

Zadejte jméno ukončovacího programu ve formátu `library(function)` a ujistěte se, že umístění uživatelského programu je zahrnuto v proměnné prostředí cesty.

Informace o tom, jak zapisovat uživatelskou proceduru kanálu v jazyce C, najdete v tématu “Kanály-uživatelské programy pro kanály systému zpráv” na stránce 380.

Použití externích tříd ukončení

Ve verzích produktu WebSphere MQ starších než verze 7.0byly poskytnuty tři třídy, které vám umožňují používat uživatelské procedury kanálu psané v jiných jazycích než Java:

- MQExternalSecurityExit, který implementuje rozhraní MQSecurityExit
- MQExternalSendExit, který implementuje rozhraní MQSendExit
- MQExternalReceiveExit, který implementuje rozhraní MQReceiveExit

Použití těchto tříd zůstává v platnosti, ale dává přednost nové metodě.

Chcete-li použít uživatelskou proceduru zabezpečení, která není napsána v jazyce Java, aplikace nejprve musela vytvořit objekt Ukončení MQExternalSecurity. Uvedená aplikace, jako parametry v konstruktoru MQExternalSecurityExit, název knihovny obsahující uživatelskou proceduru pro zabezpečení zprávy, jméno vstupního bodu pro uživatelskou proceduru zabezpečení a uživatelská data, která mají být předána uživatelské proceduře pro zabezpečení při volání. Ukončovací programy kanálu, které nejsou zapsány v Javě, byly uloženy v adresáři zobrazeném v Tabulka 88 na stránce 667.

Použití posloupnosti odeslání nebo příjmu kanálu v třídách WebSphere MQ pro jazyk Java

Třídy WebSphere MQ pro aplikaci v jazyce Java mohou používat posloupnost ukončovacích nebo přijímacích procedur kanálu, které jsou spouštěny za dědění.

Chcete-li použít posloupnost uživatelských procedur pro odesílání, aplikace může vytvořit buď seznam, nebo řetězec obsahující uživatelské procedury odeslání. Je-li použit seznam, každý prvek seznamu může mít některou z následujících hodnot:

- Instance třídy definované uživatelem, která implementuje rozhraní WMQSendExit .
- Instance třídy definované uživatelem, která implementuje rozhraní MQSendExit (pro uživatelskou proceduru odeslání v jazyce Java)
- Instance třídy uživatelské procedury MQExternalSend(pro uživatelskou proceduru odeslání, která není zapsána v jazyce Java)
- Instance třídy řetězce MQSendExit.
- Instance třídy Řetězec

Seznam nemůže obsahovat jiný seznam.

Aplikace může použít posloupnost uživatelských procedur příjmu podobným způsobem.

Je-li použit řetězec, musí se skládat z jedné nebo více definic uživatelských procedur oddělených čárkami, z nichž každá může být buď název třídy Java, nebo program v jazyce C ve formátu `library(function)`.

Aplikace pak přiřadí objekt List nebo String do pole MQEnvironment.channelSendExit před vytvořením objektu MQQueueManager .

Kontext informací předaných východům je výhradně v rámci domény východů. Je-li například uživatelská procedura jazyka Java a uživatelská procedura C zřetězená, přítomnost uživatelské procedury Java nemá žádný vliv na uživatelskou proceduru jazyka C.

Použití tříd uživatelských řetězců

Ve verzích produktu WebSphere MQ starších než verze 7.0byly k dispozici dvě třídy, které umožňují posloupnosti uživatelských procedur:

- Řetězec MQSendExit, který implementuje rozhraní MQSendExit
- Řetězec MQReceiveExit, který implementuje rozhraní MQReceiveExit

Použití těchto tříd zůstává v platnosti, ale dává přednost nové metodě. Použití tříd WebSphere MQ Classes for Java znamená, že vaše aplikace stále má závislost na `com.ibm.mq.jar` Pokud se nová sada rozhraní v balíku `com.ibm.mq.exits` používá, není žádná závislost na `com.ibm.mq.jar`.

Chcete-li použít posloupnost uživatelských procedur odeslání, aplikace vytvořila seznam objektů, kde každý objekt byl jeden z následujících objektů:

- Instance třídy definované uživatelem, která implementuje rozhraní MQSendExit (pro uživatelskou proceduru odeslání v jazyce Java)
- Instance třídy uživatelské procedury MQExternalSend(pro uživatelskou proceduru odeslání, která není zapsána v jazyce Java)
- Instance třídy řetězce MQSendExit.

Aplikace vytvořila objekt řetězce MQSendExitpředáním tohoto seznamu objektů jako parametru v konstruktoru. Aplikace by poté před vytvořením objektu MQQueueManager přiřadil objekt řetězce MQSendExitk poli MQEnvironment.sendExit .

Komprese kanálu ve třídách produktu WebSphere MQ pro prostředí Java

Komprimace dat, která proudí na kanálu, může zvýšit výkon kanálu a omezit provoz na síti. IBM WebSphere MQ classes for Java použijte kompresní funkci vestavěná do produktu IBM WebSphere MQ.

Pomocí funkce dodávané s produktem IBM WebSphere MQmůžete komprimovat data, která se přenášejí na kanály zpráv a kanály MQI, a na obou typech kanálů můžete komprimovat data záhlaví a data zprávy nezávisle na ostatních. Standardně nejsou žádná data komprimována na kanálu. Úplný popis komprese kanálu včetně toho, jak je implementován v produktu IBM WebSphere MQ, najdete v tématu Komprese dat (COMPMSG) a Komprese záhlaví (COMPHDR).

Aplikace IBM WebSphere MQ classes for Java určuje techniky, které lze použít pro kompresi dat záhlaví nebo zprávy na připojení klienta vytvořením objektu java.util.Collection . Každá kompresní technika je objekt Integer v kolekci a pořadí, ve kterém aplikace přidá techniky komprese do kolekce, je pořadí, ve kterém jsou techniky komprese vyjednané se správcem front při spuštění připojení klienta. Aplikace pak může přiřadit kolekci do pole Seznam hdrComp, pro data záhlaví nebo pole seznamu msgComppro data zprávy, ve třídě MQEnvironment. Je-li aplikace připravena, může spustit připojení klienta vytvořením objektu MQQueueManager .

Následující fragmenty kódu popisují popsaný přístup. První fragment kódu ukazuje, jak implementovat kompresi dat záhlaví:

```
Collection headerComp = new Vector();
headerComp.add(new Integer(CMQXC.MQCOMPRESS_SYSTEM));
:
MQEnvironment.hdrCompList = headerComp;
:
MQQueueManager qMgr = new MQQueueManager(QM);
```

Druhý fragment kódu ukazuje, jak implementovat kompresi dat zprávy:

```
Collection msgComp = new Vector();
msgComp.add(new Integer(CMQXC.MQCOMPRESS_RLE));
msgComp.add(new Integer(CMQXC.MQCOMPRESS_ZLIBHIGH));
:
MQEnvironment.msgCompList = msgComp;
:
MQQueueManager qMgr = new MQQueueManager(QM);
```

Ve druhém příkladu jsou techniky komprese vyjednané v pořadí RLE, pak ZLIBHIGH, když se spustí připojení klienta. Zvolená technika komprese nemůže být změněna během doby životnosti objektu MQQueueManager .

Techniky komprese pro záhlaví a data zpráv, které jsou podporovány klientem i správcem front v připojení klienta, jsou předány do uživatelské procedury kanálu jako kolekce v polích hdrCompList a msgCompList objektu MQChannelDefinition . Skutečné techniky, které se aktuálně používají pro kompresi záhlaví a dat zpráv v připojení klienta, jsou předány uživatelské proceduře kanálu v polích Komprese CurHdra komprese CurMsgobjektu MQChannelExit .

Je-li komprese použita na připojení klienta, data jsou komprimována před zpracováním a extrakcí kanálu odesílání kanálu po zpracování všech uživatelských procedur příjmu kanálu. Data předaná k odeslání a přijetí uživatelských procedur se proto nacházejí v komprimovaném stavu.

Další informace o určování technik komprese a o tom, které techniky komprese jsou k dispozici, naleznete v části Třída com.ibm.mq.MQEnvironment a Rozhraní com.ibm.mq.MQC .

Sdílení připojení TCP/IP v produktu IBM WebSphere MQ classes for Java

Je možné vytvořit více instancí kanálu MQI, aby bylo možné sdílet jedno připojení TCP/IP.

V produktu IBM WebSphere MQ classes for Javamůžete prostřednictvím proměnné MQEnvironment.sharingConversations řídit počet konverzací, které mohou sdílet jedno připojení TCP/IP.

Atribut SHARECNV je nejlepším přístupem k sdílení připojení. Proto je-li hodnota parametru SHARECNV větší než 0 použita v kombinaci s IBM WebSphere MQ classes for Java , není zaručeno, že nový požadavek na připojení bude vždy sdílet již vytvořené spojení.

Sdružování připojení do tříd produktu WebSphere MQ pro prostředí Java

Třídy WebSphere MQ pro jazyk Java umožňují rozšíření volných připojení ve fondu pro opětovné použití.

Třídy WebSphere MQ for Java poskytují dodatečnou podporu pro aplikace, které se zabývají více připojeními ke správcům front WebSphere MQ . Není-li připojení již vyžadováno, lze ji namísto zničení sloučit do fondu a později je znovu použít. To může poskytnout podstatné zvýšení výkonu pro aplikace a middleware, které se sériově připojují k libovolnému správci front.

Produkt WebSphere MQ poskytuje výchozí fond připojení. Aplikace mohou aktivovat nebo deaktivovat tento fond připojení registrací a deregistrováním tokenů prostřednictvím třídy MQEnvironment. Je-li fond aktivní, když třídy WebSphere MQ pro konstrukty jazyka Java vytvoří objekt MQQueueManager , prohledá tento výchozí fond a znovu použije jakékoli vhodné připojení. Pokud dojde k volání funkce MQQueueManager.disconnect (), je základní připojení vráceno do fondu.

Aplikace mohou alternativně vytvářet fond připojení MQSimpleConnectionManager pro konkrétní použití. Aplikace pak může buď tento fond zadat během konstrukce objektu MQQueueManager , nebo předat tento fond do prostředí MQEnvironment, aby jej bylo možné použít jako výchozí fond připojení.

Chcete-li zabránit připojení k použití příliš velkého množství prostředků, můžete omezit celkový počet připojení, která objekt správce MQSimpleConnectiondokáže zpracovat, a můžete omezit velikost fondu připojení. Nastavení limitů je užitečné, pokud existují konfliktní požadavky na připojení v rámci prostředí JVM.

Při výchozím nastavení metoda getMaxConnections () vrací hodnotu nula, což znamená, že neexistuje žádné omezení pro počet připojení, které objekt MQSimpleConnectionManager dokáže zpracovat. Limit můžete nastavit pomocí metody setMaxConnections (). Pokud jste nastavili limit a byl dosažen limit, požadavek na další připojení může způsobit, že bude vyvolána výjimka MQException s kódem příčiny MQRC_MAX_CONNS_LIMIT_REACHED.

Řízení výchozího fondu připojení ve třídách WebSphere MQ pro prostředí Java

Tento příklad ukazuje, jak používat výchozí fond připojení.

Prohlédněte si následující vzorovou aplikaci MQApp1:

```
import com.ibm.mq.*;
public class MQApp1
{
      public static void main(String[] args) throws MQException
      {
         for (int i=0; i<args.length; i++) {
            MQQueueManager qmgr=new MQQueueManager(args[i]);
            :
            : (do something with qmgr)
            :
            qmgr.disconnect();
         }
      }
}
```

Aplikace MQApp1 přijímá seznam lokálních správců front z příkazového řádku, připojuje se k jednotlivým správcům a provádí určitou operaci. Když však příkazový řádek zobrazuje vícekrát stejného správce front, je efektivnější připojit se pouze jednou a znovu použít toto připojení mnohokrát.

Třídy WebSphere MQ pro jazyk Java poskytují výchozí fond připojení, který můžete použít k provedení tohoto procesu. Chcete-li fond povolit, použijte jeden z metod MQEnvironment.addConnectionPoolToken(). Chcete-li fond zakázat, použijte volbu MQEnvironment.removeConnectionPoolToken().

Následující ukázková aplikace, MQApp2, je funkčně identická s MQApp1, ale připojuje se k jednotlivým správcům front pouze jednou.

```
import com.ibm.mq.*;
public class MQApp2
{
      public static void main(String[] args) throws MQException
      {
         MQPoolToken token=MQEnvironment.addConnectionPoolToken();

         for (int i=0; i<args.length; i++) {
            MQQueueManager qmgr=new MQQueueManager(args[i]);
            :
            : (do something with qmgr)
            :
            qmgr.disconnect();
         }
```

```
            MQEnvironment.removeConnectionPoolToken(token);

        }
}
```

První smělý řádek aktivuje výchozí fond připojení registrací objektu MQPoolToken s rozhraním MQEnvironment.

Konstruktor MQQueueManager nyní prohledá tento fond pro příslušné připojení a vytvoří připojení ke správci front pouze v případě, že nemůže najít již existující připojení. Volání qmgr.disconnect() vrátí připojení do fondu pro pozdější použití. Tato volání rozhraní API jsou stejná jako ukázková aplikace MQApp1.

Druhý zvýrazněný řádek deaktivuje výchozí fond připojení, který zničí všechna připojení správce front uložená ve fondu. To je důležité, protože v opačném případě by aplikace byla ukončena s počtem aktivních připojení správců front ve fondu. Tato situace může způsobit chyby, které se objeví v protokolech správce front.

Pokud aplikace používá tabulku definic kanálů klienta (CCDT) k připojení ke správci front, konstruktor MQQueueManager nejprve prohledá tabulku pro vhodnou definici kanálu připojení klienta. Je-li nalezen, prohledá konstruktor výchozí fond připojení pro připojení, které lze použít pro kanál. Pokud konstruktor nemůže najít vhodné připojení ve fondu, prohledá tabulku definic kanálů klienta pro další vhodnou definici kanálu připojení klienta a bude pokračovat, jak je popsáno výše. Pokud konstruktor dokončí hledání v tabulce definic kanálů klienta a nepodaří se najít žádné vhodné připojení ve fondu, konstruktor spustí druhé prohledání tabulky. Během tohoto vyhledávání se konstruktor pokusí o vytvoření nového připojení pro každou vhodnou definici kanálu připojení klienta a použije první připojení, které se bude spravovat k vytvoření.

Výchozí fond připojení ukládá maximálně deset nevyužitých připojení a udržuje nepoužívaná připojení aktivní po dobu maximálně pěti minut. Aplikace může tuto změnu změnit (podrobnosti viz “Dodání jiného fondu připojení ve třídách WebSphere MQ pro prostředí Java” na stránce 673).

Místo použití produktu MQEnvironment k zadání objektu MQPoolTokenmůže aplikace sestavit vlastní:

```
    MQPoolToken token=new MQPoolToken();
    MQEnvironment.addConnectionPoolToken(token);
```

Některé aplikace nebo dodavatelé middlewaru poskytují podtřídy MQPoolToken za účelem předávání informací do vlastního fondu připojení. Mohou být konstruována a předána do addConnectionPoolToken() takovým způsobem, aby mohly být do fondu připojení předány další informace.

Výchozí fond připojení a více komponent ve třídách produktu WebSphere MQ pro prostředí Java

Tento příklad ukazuje, jak přidat nebo odebrat položku MQPoolTokens ze statické sady registrovaných objektů MQPoolToken .

Produkt MQEnvironment uchovává statickou sadu registrovaných objektů MQPoolToken . Chcete-li přidat nebo odebrat položku MQPoolTokens z této sady, použijte následující metody:

- MQEnvironment.addConnectionPoolToken()
- MQEnvironment.removeConnectionPoolToken()

Aplikace se může skládat z mnoha komponent, které existují nezávisle a vykonávají práci pomocí správce front. V takové aplikaci by každá komponenta měla přidat objekt MQPoolToken do sady MQEnvironment po dobu jeho životnosti.

Například vzorová aplikace MQApp3 vytvoří deset podprocesů a spustí každou z nich. Každý podproces registruje svůj vlastní MQPoolToken, čeká po určitou dobu a potom se připojí ke správci front. Jakmile se podproces odpojí, odebere svůj vlastní MQPoolToken.

Výchozí fond připojení zůstane aktivní, je-li v sadě MQPoolTokensnastaven alespoň jeden token, takže zůstane aktivní po dobu trvání této aplikace. Aplikace nemusí udržovat hlavní objekt v celkovém řízení podprocesů.

```
import com.ibm.mq.*;
public class MQApp3
{
      public static void main(String[] args)
      {
         for (int i=0; i<10; i++) {
            MQApp3_Thread thread=new MQApp3_Thread(i*60000);
            thread.start();
         }
      }
}

class MQApp3_Thread extends Thread
{
      long time;

      public MQApp3_Thread(long time)
      {
         this.time=time;
      }

      public synchronized void run()
      {
         MQPoolToken token=MQEnvironment.addConnectionPoolToken();
         try {
            wait(time);
            MQQueueManager qmgr=new MQQueueManager("my.qmgr.1");
            :
            : (do something with qmgr)
            :
            qmgr.disconnect();
         }
         catch (MQException mqe) {System.err.println("Error occurred!");}
         catch (InterruptedException ie) {}

         MQEnvironment.removeConnectionPoolToken(token);
      }
}
```

Dodání jiného fondu připojení ve třídách WebSphere MQ pro prostředí Java

Tento příklad ukazuje, jak lze použít třídu **com.ibm.mq.MQSimpleConnectionManager** k zajištění jiného fondu připojení.

Tato třída poskytuje základní prostředky pro použití fondu připojení a aplikace mohou tuto třídu používat k přizpůsobení chování fondu.

Jakmile je vytvořena instance, lze v konstruktoru MQQueueManager zadat správce MQSimpleConnectionManager. Správce MQSimpleConnectionpoté spravuje připojení, které je základem konstruovaného objektu MQQueueManager. Pokud správce MQSimpleConnectionobsahuje vhodné připojení ve fondu, je toto připojení znovu použito a vráceno do správce MQSimpleConnectionza voláním funkce MQQueueManager.disconnect ().

Toto chování demonstruje následující fragment kódu:

```
MQSimpleConnectionManager myConnMan=new MQSimpleConnectionManager();
myConnMan.setActive(MQSimpleConnectionManager.MODE_ACTIVE);
MQQueueManager qmgr=new MQQueueManager("my.qmgr.1", myConnMan);
:
: (do something with qmgr)
:
qmgr.disconnect();

MQQueueManager qmgr2=new MQQueueManager("my.qmgr.1", myConnMan);
:
: (do something with qmgr2)
:
qmgr2.disconnect();
myConnMan.setActive(MQSimpleConnectionManager.MODE_INACTIVE);
```

Připojení, které je vytvořeno během prvního konstruktoru MQQueueManager , je uloženo v adresáři myConnMan po volání qmgr.disconnect(). Připojení je poté znovu použito při druhém volání konstruktoru MQQueueManager .

Druhý řádek umožňuje správce MQSimpleConnectionManager. Poslední řádek vypne správce MQSimpleConnectiona zničí veškerá spojení, která se nacházejí ve fondu. Správce MQSimpleConnectionje ve výchozím nastavení v MODE_AUTO, který je popsán později v této sekci.

Správce MQSimpleConnectionpřiděluje připojení k nejnověji využívaným základům a likviduje připojení na základě nejdéle nepoužitého základu. Při výchozím nastavení je připojení zničeno, pokud nebylo použito pět minut, nebo pokud ve fondu existuje více než deset nepoužívaných připojení. Tyto hodnoty můžete změnit voláním MQSimpleConnectionManager.setTimeout().

Můžete také nastavit produkt MQSimpleConnectionManager pro použití jako výchozí fond připojení, který má být použit v případě, že není v konstruktoru MQQueueManager zadán žádný správce Connection Manager.

Následující aplikace demonstruje toto:

```
import com.ibm.mq.*;
public class MQApp4
{
      public static void main(String []args)
      {
         MQSimpleConnectionManager myConnMan=new MQSimpleConnectionManager();
         myConnMan.setActive(MQSimpleConnectionManager.MODE_AUTO);
         myConnMan.setTimeout(3600000);
         myConnMan.setMaxConnections(75);
         myConnMan.setMaxUnusedConnections(50);
         MQEnvironment.setDefaultConnectionManager(myConnMan);
         MQApp3.main(args);
      }
}
```

Tučné čáry vytvářejí a konfigurují objekt MQSimpleConnectionManager. Konfigurace provede následující akce:

- Ukončí spojení, která se nepoužívají po dobu jedné hodiny.
- Omezuje počet připojení spravovaných produktem myConnMan až 75.
- Omezuje počet nevyužitých připojení ve fondu na 50.
- Nastavuje MODE_AUTO, což je výchozí nastavení. To znamená, že fond je aktivní pouze v případě, že se jedná o výchozího správce připojení a v sadě MQPoolTokens , které má v držení MQEnvironment, je alespoň jeden token.

Nový správce MQSimpleConnectionje poté nastaven jako výchozí správce připojení.

V posledním řádku volá aplikace funkci MQApp3.main(). Tím se spustí počet podprocesů, kde každý podproces používá nezávisle produkt WebSphere MQ . Tyto podprocesy používají myConnMan, když navazují spojení.

Dodání vlastního produktu ConnectionManager pro třídy WebSphere MQ pro prostředí Java

Třídy WebSphere MQ pro prostředí Java poskytují částečnou implementaci architektury Java EE Connector Architecture, která umožňuje použití implementací javax.resource.spi.ConnectionManager .

Aplikace a poskytovatelé middlewaru mohou poskytovat alternativní implementace fondů připojení. Třídy WebSphere MQ for Java poskytují částečnou implementaci architektury Java EE Connector Architecture. Implementace **javax.resource.spi.ConnectionManager** mohou být buď použita jako výchozí správce Connection Manager, nebo mohou být určena v konstruktoru MQQueueManager .

Třídy WebSphere MQ pro jazyk Java jsou v souladu se smlouvou o správě připojení architektury Java EE Connector Architecture. Přečtěte si tuto část ve spojení se smlouvou o správě připojení architektury Java EE Connector Architecture (viz webový server Sun Java na adrese https://java.sun.com).

Rozhraní ConnectionManager definuje pouze jednu metodu:

```
package javax.resource.spi;
public interface ConnectionManager {
      Object allocateConnection(ManagedConnectionFactory mcf,
                                ConnectionRequestInfo cxRequestInfo);
}
```

Konstruktor MQQueueManager volá allocateConnection na příslušné ConnectionManager. Propojí odpovídající implementace továrny ManagedConnectionFactory a ConnectionRequestjako parametry pro popis požadovaného připojení.

Volba ConnectionManager prohledá svůj fond pro objekt javax.resource.spi.ManagedConnection , který byl vytvořen s identickými objekty ManagedConnectionFactory a ConnectionRequestInfo. Pokud ConnectionManager najde vhodné objekty ManagedConnection , vytvoří java.util.Set , která obsahuje kandidáta ManagedConnections. Pak ConnectionManager volá následující:

```
ManagedConnection mc=mcf.matchManagedConnections(connectionSet, subject,
cxRequestInfo);
```

The WebSphere MQ implementation of ManagedConnectionFactory ignores the subject parameter. Tato metoda vybere a vrátí odpovídající objekt ManagedConnection ze sady nebo vrátí hodnotu null, pokud nenajde vhodné ManagedConnection. Pokud ve fondu není vhodné ManagedConnection , může ConnectionManager vytvořit jeden pomocí:

```
ManagedConnection mc=mcf.createManagedConnection(subject, cxRequestInfo);
```

Parametr předmětu je opět ignorován. Tato metoda se připojuje ke správci front produktu WebSphere MQ a vrací implementaci javax.resource.spi.ManagedConnection , která představuje nově kované připojení. Jakmile produkt ConnectionManager získá ManagedConnection (buď z fondu nebo nově vytvořeného), vytvoří manipulátor připojení pomocí:

```
Object handle=mc.getConnection(subject, cxRequestInfo);
```

Tento manipulátor připojení může být vrácen z příkazu allocateConnection().

Hodnota ConnectionManager musí zaregistrovat zájem o ManagedConnection prostřednictvím:

```
mc.addConnectionEventListener()
```

Modul listener ConnectionEventje upozorněn, dojde-li k závažné chybě v připojení, nebo při volání funkce MQQueueManager.disconnect (). Je-li volána funkce MQQueueManager.disconnect (), může modul listener ConnectionEventprovést jednu z následujících akcí:

- Resetuje ManagedConnection pomocí volání mc.cleanup(), pak vrátí objekt ManagedConnection do fondu
- Zlikvidovat ManagedConnection pomocí volání mc.destroy()

Je-li ConnectionManager výchozím správcem ConnectionManager, může také registrovat zájem o stav sady MQPoolTokensve spravované MQEnvironment. Chcete-li tak učinit, nejprve vytvořte objekt MQPoolServices a poté zaregistrujte objekt MQPoolServicesEventListener s objektem MQPoolServices :

```
MQPoolServices mqps=new MQPoolServices();
mqps.addMQPoolServicesEventListener(listener);
```

Listener je upozorněn, když je objekt MQPoolToken přidán nebo odebrán ze sady, nebo když se změní výchozí ConnectionManager . Objekt MQPoolServices také poskytuje způsob dotazování na aktuální velikost sady MQPoolTokens.

Koordinace JTA/JDBC používající třídy WebSphere MQ pro prostředí Java

Třídy WebSphere MQ pro jazyk Java podporují metodu MQQueueManager.begin (), která umožňuje produktu WebSphere MQ vystupovat jako koordinátor pro databázi, která poskytuje ovladač vyhovující standardu JDBC typu 2 nebo JDBC typu 4.

Tato podpora není k dispozici na všech platformách. Chcete-li zkontrolovat, které platformy podporují koordinaci JDBC , prohlédněte si téma https://www.ibm.com/software/integration/wmq/requirements/.

Chcete-li použít podporu XA-JTA, musíte použít speciální knihovnu přepínačů JTA. Způsob použití této knihovny se liší v závislosti na tom, zda používáte systém Windows nebo jeden z jiných platforem.

Konfigurace koordinace JTA/JDBC na serveru Windows

Knihovna XA je dodávána jako knihovna DLL s názvem ve formátu `jdbcxxx.dll`.

V 7.5.0.7 Dodané `jdbcora12.dll` poskytuje kompatibilitu s Oracle 12C, pro instalaci serveru IBM WebSphere MQ Windows .

V systémech Windows je knihovna XA dodávána jako úplná knihovna DLL. Název této knihovny DLL je `jdbcxxx.dll` , kde `xxx` označuje databázi, pro kterou byla knihovna přepínače kompilována. Tato knihovna se nachází v adresáři `java\lib\jdbc` nebo `java\lib64\jdbc` vašich tříd produktu IBM WebSphere MQ pro instalaci produktu Java . Musíte deklarovat knihovnu XA, která je také popsána jako zaváděcí soubor přepínače, do správce front. Použijte IBM WebSphere MQ Explorer. V panelu vlastností správce front určete podrobnosti o souboru načtení přepínače v panelu vlastností správce front. Je třeba zadat pouze název knihovny. Příklad:

V případě databáze Db2 nastavte pole SwitchFile na: `dbcdb2`

Pro databázi Oracle nastavte pole SwitchFile na: `jdbcora`

Konfigurace koordinace JTA/JDBC na platformách jiných než Windows

Jsou dodány soubory objektů. Propojte příslušný soubor s použitím dodaného souboru Makefile a deklarujte jej pro správce front pomocí konfiguračního souboru.

Pro každý systém správy databází produkt WebSphere MQ poskytuje dva objektové soubory. Chcete-li vytvořit 32bitovou knihovnu přepínačů, musíte propojit jeden soubor objektu a propojit jiný objektový soubor a vytvořit 64bitovou knihovnu přepínačů. Pro databázi DB2je název každého objektového souboru jdbcdb2.o a pro Oracleje název každého objektového souboru jdbcora.o.

Každý soubor objektu je třeba propojit s použitím příslušného souboru Makefile dodávaného s produktem WebSphere MQ. Knihovna přepínače vyžaduje jiné knihovny, které mohou být uloženy v různých lokalitách na různých systémech. Knihovna přepínačů však nemůže k nalezení těchto knihoven použít proměnnou prostředí cesty ke knihovně, protože knihovna přepínačů je načtena správcem front, který je spuštěn v prostředí setuid. Dodaný soubor Makefile proto zaručuje, že knihovna přepínačů obsahuje úplné názvy cest těchto knihoven.

Chcete-li vytvořit knihovnu přepínačů, zadejte příkaz **make** s následujícím formátem. Chcete-li vytvořit 32bitovou knihovnu přepínačů, zadejte příkaz v adresáři /java/lib/jdbc v instalaci produktu WebSphere MQ . Chcete-li vytvořit 64bitovou knihovnu přepínačů, zadejte příkaz v adresáři /java/lib64/jdbc .

```
make DBMS
```

kde *DBMS* je systém správy databází, pro který vytváříte knihovnu přepínačů. Platné hodnoty jsou db2 pro DB2 a `oracle` pro Oracle.

Zde je uveden příklad příkazu **make** :

```
make db2
```

Všimněte si následujících bodů:

- Chcete-li spustit 32bitové aplikace, musíte pro každý systém správy databází, který používáte, vytvořit 32bitovou a 64bitovou knihovnu přepínačů. Chcete-li spustit 64bitové aplikace, musíte vytvořit pouze 64bitovou knihovnu přepínačů. Pro databázi DB2je název každé knihovny přepínače jdbcdb2 a v případě

Oracleje název každé knihovny přepínače jdbcora. Soubory Makefile zajišťují, aby 32bitové a 64bitové knihovny přepínačů byly uloženy v různých adresářích produktu WebSphere MQ . 32bitovou knihovnu přepínačů je uložena v adresáři /java/lib/jdbc a 64bitová knihovna s přepínačem je uložena v adresáři / java/lib64/jdbc .

- Protože můžete produkt Oracle nainstalovat kdekoli na systému, soubory Makefile používají proměnnou prostředí ORACLE_HOME k vyhledání umístění, kde je nainstalována databáze Oracle .

Po vytvoření knihoven přepínače pro produkt DB2, Oraclenebo obojí je třeba je deklarovat pro správce front. Pokud konfigurační soubor správce front (qm.ini) již obsahuje stanzy XAResourceManager pro databáze DB2 nebo Oracle , musíte nahradit položku SwitchFile v každém oddílu jedním z následujících způsobů:

Pro databázi DB2

```
SwitchFile=jdbcdb2
```

Pro databázi Oracle

```
SwitchFile=jdbcora
```

Neuvádějte plně kvalifikovaný název cesty buď 32bitové, nebo 64bitové knihovny přepínačů. Uveďte pouze název knihovny.

Pokud konfigurační soubor správce front již neobsahuje sekce XAResourceManager pro databáze DB2 nebo Oracle , nebo pokud chcete přidat další sekce XAResourceManager , prohlédněte si Administrace , kde naleznete informace o tom, jak sestavit sekci XAResourceManager . Avšak každý záznam SwitchFile v nové stanze XAResourceManager musí být přesně tak, jak je popsáno dříve pro databázi DB2 nebo Oracle . Musíte také zahrnout položku `ThreadOfControl=PROCESS`.

Po aktualizaci konfiguračního souboru správce front a ujistit se, že byly nastaveny všechny příslušné proměnné prostředí databáze, můžete správce front restartovat.

Použití koordinace JTA/JDBC

Okódovali jste volání rozhraní API jako v dodaném příkladu.

Základní posloupnost volání rozhraní API pro uživatelskou aplikaci je:

```
qMgr = new MQQueueManager("QM1")
Connection con = qMgr.getJDBCConnection( xads );
qMgr.begin()

< Perform MQ and DB operations to be grouped in a unit of work >

qMgr.commit() or qMgr.backout();
con.close()
qMgr.disconnect()
```

`xads` v rámci volání getJDBCConnection je implementace rozhraní XADataSource specifická pro databázi, která definuje podrobnosti o databázi, k níž se má připojit. Informace o tom, jak vytvořit příslušný objekt XADataSource pro předání do getJDBCConnection, najdete v dokumentaci k vaší databázi.

Dále je třeba aktualizovat cestu ke třídám s použitím vhodných souborů JAR specifických pro databázi pro provádění práce s produktem JDBC .

Pokud se musíte připojit k více databázím, musíte několikrát zavolat getJDBCConnection k provedení transakce přes několik různých připojení.

Existují dvě formy getJDBCConnection, které odrážejí dvě formy XADataSource.getXAConnection:

```
public java.sql.Connection getJDBCConnection(javax.sql.XADataSource xads)
  throws MQException, SQLException, Exception

public java.sql.Connection getJDBCConnection(XADataSource dataSource,
                                             String userid, String password)
  throws MQException, SQLException, Exception
```

Tyto metody deklarují výjimku ve svých klauzulích throws, aby se vyvarovali problémů s ověřovatelem prostředí JVM pro zákazníky, kteří nepoužívají funkce JTA. Skutečná vyvolaná výjimka je javax.transaction.xa.XAException , která vyžaduje, aby byl soubor jta.jar přidán do cesty ke třídě pro programy, které na ni dříve nevyžadovaly.

Chcete-li použít podporu JTA/JDBC , musíte do své aplikace zahrnout následující příkaz:

```
MQEnvironment.properties.put(CMQC.THREAD_AFFINITY_PROPERTY, new Boolean(true));
```

Známé problémy a omezení s koordinací JTA/JDBC

Existují určité problémy a omezení podpory JTA/JDBC , některé v závislosti na systému správy databází, které se používají.

Vzhledem k tomu, že tato podpora volá ovladače JDBC , může mít implementace těchto ovladačů JDBC významný vliv na chování systému. Testované ovladače JDBC se budou chovat jinak, je-li ukončena činnost databáze, zatímco je spuštěna aplikace. Volba **Vždy** se vyhnout náhlému ukončení činnosti databáze v době, kdy jsou k dispozici aplikace, které drží otevřená spojení.

Více oddílů XAResourceManager
Použití více než jednoho objektu stanza XAResourceManager v konfiguračním souboru správce front qm.ininení podporováno. Jakákoli sekce XAResourceManager jiná než první je ignorována.

DB2

Někdy DB2 vrátí chybu SQL0805N . Tento problém lze vyřešit pomocí následujícího příkazu příkazového procesoru:

```
DB2 bind @db2cli.lst blocking all grant public
```

Další informace naleznete v dokumentaci produktu DB2 .

Objekt stanza XAResourceManager musí být nakonfigurován pro použití ThreadOfControl=PROCESS. Pro databázi DB2 verze 8.1 a vyšší to neodpovídá výchozímu podprocesu řídicího nastavení pro produkt DB2, takže toc=p musí být určeno v otevřeném řetězci XA. Příklad 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 nezabrání aplikacím Java, které používají koordinaci JTA/JDBC , aby byly s podporou více podprocesů.

Oracle
Volání metody JDBC Connection.close() po funkci MQQueueManager.disconnect () generuje výjimku SQLException. Buď volejte Connection.close() před MQQueueManager.disconnect (), nebo vynechte volání na Connection.close().

Podpora zabezpečení Secure Sockets Layer (SSL) ve třídách WebSphere MQ pro prostředí Java

Třídy WebSphere MQ pro klientské aplikace jazyka Java podporují šifrování SSL (Secure Sockets Layer). Chcete-li používat šifrování SSL, je třeba poskytovatele prostředí JSSE.

Třídy WebSphere MQ pro klientské aplikace Java používající funkci TRANSPORT (TRANSPORT) podporují šifrování SSL (Secure Sockets Layer). SSL poskytuje komunikační šifrování, ověření a integritu zpráv. Obvykle se používá k zabezpečení komunikace mezi dvěma rovnocennými partnery na Internetu nebo v rámci intranetu.

WebSphere MQ Classes for Java používá k ošetření šifrování SSL JSSE (Java Secure Socket Extension), a proto vyžaduje poskytovatele JSSE. Prostředí JVM JSE v1.4 má zabudovaný poskytovatel JSSE.

Podrobnosti o tom, jak spravovat a ukládat certifikáty se mohou lišit od poskytovatele k poskytovateli. Informace o tomto tématu naleznete v dokumentaci k poskytovateli JSSE.

V tomto oddílu se předpokládá, že poskytovatel JSSE je správně nainstalován a nakonfigurován a že byly nainstalovány a zpřístupněny vhodné certifikáty pro poskytovatele JSSE.

Pokud vaše aplikace klienta WebSphere MQ pro klientskou aplikaci Java používá tabulku definic kanálů klienta (CCDT) k připojení ke správci front, přečtěte si téma "Použití tabulky definic kanálů klienta s IBM WebSphere MQ classes for Java" na stránce 650.

Povolení zabezpečení SSL v produktu IBM WebSphere MQ classes for Java

Chcete-li povolit zabezpečení SSL, zadejte sadu CipherSuite. Existují dva způsoby, jak určit sadu CipherSuite.

SSL je podporováno pouze pro připojení klienta. Chcete-li povolit zabezpečení SSL, je třeba určit volbu CipherSuite , která má být použita při komunikaci se správcem front, a tato sada CipherSuite musí odpovídat sadě CipherSpec na cílovém kanálu. Kromě toho musí poskytovatel JSSE podporovat název CipherSuite . Hodnota CipherSuites se však liší od specifikace CipherSpecs a mají tedy různé názvy. Produkt "SSL CipherSpecs a CipherSuites v třídách WebSphere MQ pro prostředí Java" na stránce 683 obsahuje tabulku CipherSpecs podporované produktem IBM WebSphere MQ pro jejich ekvivalent k rovnocennému souboru CipherSuites , jak je známo jako rozšíření JSSE.

Chcete-li povolit zabezpečení SSL, zadejte sadu CipherSuite pomocí proměnné statického člena sady sslCipherpro prostředí MQEnvironment. Následující příklad se připojuje ke kanálu SVRCONN s názvem SECURE.SVRCONN.CHANNEL, která byla nastavena tak, aby vyžadovala zabezpečení SSL se sadou CipherSpec objektu RC4_MD5_EXPORT:

```
MQEnvironment.hostname       = "your_hostname";
MQEnvironment.channel        = "SECURE.SVRCONN.CHANNEL";
MQEnvironment.sslCipherSuite = "SSL_RSA_EXPORT_WITH_RC4_40_MD5";
MQQueueManager qmgr = new MQQueueManager("your_Q_manager");
```

Ačkoli má kanál CipherSpec RC4_MD5_EXPORT, aplikace Java musí určovat sadu CipherSuite pro SSL_RSA_EXPORT_WITH_RC4_40_MD5. Seznam mapování mezi CipherSpecs a CipherSuitesviz "SSL CipherSpecs a CipherSuites v třídách WebSphere MQ pro prostředí Java" na stránce 683 .

Aplikace může také určit sadu CipherSuite nastavením vlastnosti prostředí CMQC.SSL_CIPHER_SUITE_PROPERTY.

Případně můžete použít tabulku CCDT (Client Channel Definition Table). Další informace viz "Použití tabulky definic kanálů klienta s IBM WebSphere MQ classes for Java" na stránce 650

If you require a client connection to use a CipherSuite that is supported by the IBM Java JSSE FIPS provider (IBMJSSEFIPS), an application can set the sslFipsRequired field in the MQEnvironment class to `true`. Alternativně může aplikace nastavit vlastnost prostředí CMQC.SSL_FIPS_REQUIRED_PROPERTY. Výchozí hodnota je `false`, což znamená, že připojení klienta může použít jakoukoli sadu CipherSuite , kterou podporuje produkt IBM WebSphere MQ.

Pokud aplikace používá více než jedno připojení klienta, hodnota pole sslFips, která se použije, když aplikace vytvoří první připojení klienta, určuje hodnotu, která se použije, když aplikace vytvoří jakékoli následné připojení klienta. Proto, když aplikace vytvoří následné připojení klienta, hodnota požadovaného pole sslFipsje ignorována. Chcete-li použít jinou hodnotu pro pole Vyžadováno sslFips, musíte aplikaci restartovat.

Pro úspěšné připojení pomocí protokolu SSL musí být úložiště údajů o důvěryhodnosti JSSE nastaveno na kořenové certifikáty certifikační autority, z nichž lze ověřit certifikát prezentovaný správcem front. Podobně, je-li vlastnost SSLClientAuth v kanálu SVRCONN nastavena na hodnotu MQSSL_CLIENT_AUTH_REQUIRED, musí úložiště klíčů JSSE obsahovat identifikační certifikát, kterému správce front důvěřuje.

Související odkazy

Federální standardy zpracování informací (FIPS) pro UNIX, Linux a Windows

Použití rozlišujícího názvu správce front v produktu IBM WebSphere MQ classes for Java

Správce front identifikuje sám sebe pomocí certifikátu SSL, který obsahuje rozlišující název (DN). Klientská aplikace IBM WebSphere MQ classes for Java může tento rozlišující název použít k ujištění, že komunikuje se správným správcem front.

Vzorek DN se zadává pomocí proměnné názvu sslPeerproměnné MQEnvironment. Například nastavení:

```
MQEnvironment.sslPeerName = "CN=QMGR.*, OU=IBM, OU=WEBSPHERE";
```

umožňuje úspěšné připojení k úspěchu pouze v případě, že správce front předloží certifikát s názvem Common Name začínajícím QMGR., a alespoň dva názvy organizačních jednotek, první z nich musí být IBM a druhý WebSphere.

Je-li nastaven název sslPeer, připojení bude úspěšná pouze tehdy, je-li nastavena na platný vzor a správce front předloží odpovídající certifikát.

Aplikace může také určit rozlišující název správce front nastavením vlastnosti prostředí CMQC.SSL_PEER_NAME_PROPERTY. Další informace o rozlišujících názvech naleznete v tématu Rozlišovací jména.

Použití seznamů odvolaných certifikátů v produktu IBM WebSphere MQ classes for Java

Určete seznamy odvolaných certifikátů, které mají být použity, prostřednictvím třídy java.security.cert.CertStore . IBM WebSphere MQ classes for Java pak zkontroluje certifikáty proti uvedenému seznamu CRL.

Seznam zrušených certifikátů (CRL) je sada certifikátů, které byly odvolány, a to buď vydávající certifikační autoritou, nebo místní organizací. Seznamy CRL jsou obvykle hostovány na serverech LDAP. Při použití Java 2 v1.4může být server CRL uveden v době připojení a certifikát představený správcem front je před povolením připojení zkontrolován vůči seznamu odvolaných certifikátů. Další informace o seznamech zrušených certifikátů a o produktu IBM WebSphere MQnaleznete v tématu Práce s seznamy odvolaných certifikátů a seznamy odvolaných certifikátů a Přístup k seznamům CRL a ARL s třídami produktu WebSphere MQ pro jazyk Java a třídami produktu WebSphere MQ pro systém JMS.

Poznámka: Chcete-li produkt CertStore úspěšně používat s názvem CRL hostovaným na serveru LDAP, ujistěte se, že vaše sada SDK (Software Development Kit) produktu Java je kompatibilní se seznamem CRL. Některé sady SDK vyžadují, aby seznam CRL odpovídal RFC 2587, které definuje schéma pro protokol LDAP v2. Většina serverů LDAP v3 používá místo toho RFC 2256.

Seznamy CRL, které se mají použít, jsou určeny prostřednictvím třídy java.security.cert.CertStore . Podrobné informace o tom, jak získat instance položky CertStore, naleznete v dokumentaci k této třídě. Chcete-li vytvořit úložiště CertStore na základě serveru LDAP, nejprve vytvořte instanci parametrů LDAPCertStoreinicializovanou s nastavením serveru a portu, které se mají použít. Příklad:

```
import java.security.cert.*;
CertStoreParameters csp = new LDAPCertStoreParameters("crl_server", 389);
```

Po vytvoření instance Parameters CertStorepoužijte statický konstruktor na CertStore k vytvoření CertStore typu LDAP:

```
CertStore cs = CertStore.getInstance("LDAP", csp);
```

Podporovány jsou také další typy CertStore (například Kolekce). Pro poskytnutí redundance jsou k dispozici pouze některé servery CRL, které mají identické informace CRL. Máte-li objekt CertStore pro každý z těchto serverů CRL, umístěte je do vhodné kolekce. Následující příklad ukazuje objekty CertStore umístěné v ArrayList:

```
import java.util.ArrayList;
```

```
Collection crls = new ArrayList();
crls.add(cs);
```

Tato kolekce může být nastavena do statické proměnné MQEnvironment sslCert, než se připojí k povolení kontroly CRL:

```
MQEnvironment.sslCertStores = crls;
```

Certifikát, který předkládá správce front při nastavení připojení, je ověřen následujícím způsobem:

1. První objekt CertStore v kolekci identifikovaný pomocí úložišť sslCertse používá k identifikaci serveru CRL.
2. Došlo k pokusu o kontaktování serveru CRL.
3. Je-li pokus úspěšný, prohledá se server na shodu certifikátu.
   a. Pokud má být certifikát odvolán, proces vyhledávání skončil a žádost o připojení selže s kódem příčiny MQRC_SSL_CERTIFICATE_REVOKED.
   b. Pokud certifikát nebyl nalezen, je proces vyhledávání znovu spuštěn a připojení je povoleno pokračovat.
4. Je-li pokus o kontaktování serveru neúspěšný, bude použit další objekt CertStore pro identifikaci serveru CRL a proces se opakuje z kroku 2.

   Pokud se jednalo o poslední položku CertStore v kolekci, nebo pokud kolekce neobsahuje žádné objekty CertStore , došlo k selhání procesu vyhledávání a požadavek na připojení se nezdařil s kódem příčiny MQRC_SSL_CERT_STORE_ERROR.

Objekt kolekce určuje pořadí, ve kterém jsou použita položka CertStores .

Kolekce CertStores může být také nastavena pomocí CMQC.SSL_CERT_STORE_PROPERTY. Tato vlastnost také umožňuje zadat jednu položku CertStore bez členství v kolekci.

Je-li úložiště sslCertnastaveno na hodnotu null, neprovádí se žádná kontrola CRL. Tato vlastnost je ignorována, není-li nastavena sada sslCipherSuite.

Opětovné dohadování tajného klíče ve třídách WebSphere MQ pro prostředí Java

Třídy WebSphere MQ pro aplikaci klienta Java mohou řídit, kdy se bude znovu vyjednávat tajný klíč, který se používá pro šifrování na připojení klienta, z hlediska celkového počtu odeslaných a přijatých bajtů.

Aplikace to může provést jedním z následujících způsobů: Pokud aplikace používá více než jeden z těchto způsobů, použijí se obvyklá pravidla přednosti.

- Nastavením pole Počet sslResetve třídě MQEnvironment.
- Nastavením vlastnosti prostředí MQC.SSL_RESET_COUNT_PROPERTY v objektu Hashtable. Aplikace pak přiřadí tabulku hashtable do pole `properties` ve třídě MQEnvironment nebo předá hašovací tabulku objektu MQQueueManager do svého konstruktoru.

Hodnota pole sslResetCount nebo vlastnost prostředí MQC.SSL_RESET_COUNT_PROPERTY představuje celkový počet bajtů odeslaných a přijatých kódem klienta WebSphere MQ pro kód klienta Java před opětovným získáním tajného klíče. Počet odeslaných bajtů je číslo před šifrováním a počet přijatých bajtů je číslo po dešifrování. Počet bajtů zahrnuje také řídicí informace odeslané a přijaté třídami WebSphere MQ pro klienta Java.

Pokud je počet obnovení nulový, což je výchozí hodnota, tajný klíč není nikdy znovu vyjednáván. Počet obnovení je ignorován, pokud není zadán parametr CipherSuite .

Dodání přizpůsobené SSLSocketFactory v produktu IBM WebSphere MQ classes for Java

Pokud používáte přizpůsobenou továrnu JSSE Socket Factory, nastavte vlastnost MQEnvironment.sslSocketFactory na přizpůsobenou továrnu objektů. Podrobnosti se liší mezi různými implementacemi JSSE.

Různé implementace JSSE mohou poskytovat různé funkce. Například specializovaná implementace JSSE může umožnit konfiguraci konkrétního modelu šifrovacího hardwaru. Kromě toho někteří poskytovatelé JSSE umožňují přizpůsobení úložiště klíčů a úložiště údajů o důvěryhodnosti podle programu nebo umožňují změnu výběru certifikátu identity z úložiště klíčů. V prostředí JSSE jsou všechna tato přizpůsobení abstrahována ve třídě továrny javax.net.ssl.SSLSocketFactory.

Podrobné informace o tom, jak vytvořit přizpůsobenou implementaci SSLSocketFactory , najdete v dokumentaci k prostředí JSSE. Podrobnosti se liší od poskytovatele k poskytovateli, ale typická posloupnost kroků může být:

1. Vytvoření objektu SSLContext s použitím statické metody v SSLContext
2. Inicializujte tento SSLContext s odpovídajícími implementacemi KeyManager a TrustManager (vytvořenými z jejich vlastních továrních tříd).
3. Vytvoření objektu SSLSocketFactory z kontextu SSLContext

Máte-li objekt SSLSocketFactory , nastavte objekt MQEnvironment.sslSocketFactory na upravený objekt továrny. Příklad:

```
javax.net.ssl.SSLSocketFactory sf = sslContext.getSocketFactory();
MQEnvironment.sslSocketFactory = sf;
```

Produkt IBM WebSphere MQ classes for Java používá tento parametr SSLSocketFactory pro připojení ke správci front produktu IBM WebSphere MQ . Tuto vlastnost lze také nastavit pomocí parametru CMQC.SSL_SOCKET_FACTORY_PROPERTY. Je-li parametr sslSocketFactory nastaven na hodnotu null, použije se výchozí hodnota SSLSocketFactory prostředí JVM. Tato vlastnost je ignorována, není-li nastavena sada sslCipherSuite.

Použijete-li vlastní SSLSocketFactories, zvažte vliv sdílení připojení TCP/IP. Je-li sdílení připojení možné, není požadován nový soket pro SSLSocketFactory , a to i v případě, že by se soket vytvořil jiným způsobem v kontextu následného požadavku na připojení. Pokud má být například při následném připojení zobrazen jiný certifikát klienta, pak sdílení připojení nesmí být povoleno.

Provedení změn úložiště klíčů JSSE nebo úložiště údajů o důvěryhodnosti v produktu WebSphere MQ classes for Java

Změníte-li úložiště klíčů nebo úložiště údajů o důvěryhodnosti prostředí JSSE, musíte provést určité akce, aby se změny projevily.

Pokud změníte obsah úložiště klíčů nebo úložiště údajů o důvěryhodnosti prostředí JSSE nebo změníte umístění souboru úložiště klíčů nebo úložiště údajů o důvěryhodnosti, třídy produktu WebSphere MQ pro aplikace v jazyce Java, které jsou spuštěny v daném okamžiku, nebudou automaticky vyzvedne změny. Aby se změny projevily, musí být provedeny následující akce:

- Aplikace musí zavřít všechna svá připojení a zničit veškerá nepoužívaná připojení ve fondech připojení.
- Pokud poskytovatel JSSE ukládá informace z úložiště klíčů a úložiště údajů o důvěryhodnosti do mezipaměti, musí být tyto informace aktualizovány.

Po provedení těchto akcí mohou aplikace znovu vytvořit svá připojení.

V závislosti na návrhu aplikací a na funkci poskytované vaším poskytovatelem JSSE může být možné provést tyto akce bez zastavení a restartování aplikací. Avšak zastavení a restartování aplikací může být nejjednodušším řešením.

Ošetření chyb při použití SSL s třídami WebSphere MQ pro Java

Při připojování ke správci front s použitím zabezpečení SSL může produkt WebSphere MQ pro jazyk Java vydávat kódy kódů příčiny.

Ty jsou vysvětleny v následujícím seznamu:

MQRC_SSL_NOT_ALLOWED
Vlastnost sady sslCipherbyla nastavena, ale bylo použito připojení vazeb. SSL podporuje pouze připojení klienta.

CHYBA MQRC_JSSE_ERROR
Poskytovatel JSSE ohlásil chybu, kterou nebylo možné zpracovat pomocí produktu WebSphere MQ. Příčinou může být problém s konfigurací s podporou JSSE, nebo proto, že certifikát předložený správcem front nelze ověřit. Výjimka produkovaná JSSE může být načtena pomocí metody getCause() příkazu MQException.

CHYBA MQRC_SSL_INITIALIZATION_ERROR
Bylo vydáno volání MQCONN nebo MQCONNX s uvedenými volbami konfigurace SSL, ale během inicializace prostředí SSL se vyskytla chyba.

NESROVNALOST MQRC_SSL_PEER_NAME_
Vzorek DN zadaný ve vlastnosti názvu sslPeerneodpovídal rozlišujícímu názvu představenému správcem front.

CHYBA MQRC_SSL_PEER_NAME_ERROR
Vzorek DN zadaný ve vlastnosti sslPeerName nebyl platný.

MQRC_UNSUPPORTED_CIPHER_SUITE
Poskytovatel JSSE nerozeznal sadu CipherSuite uvedenou v sadě sslCipherSuite. Úplný seznam CipherSuites podporovaný poskytovatelem JSSE může být získán pomocí programu pomocí metody SSLSocketFactory.getSupportedCipherSuites(). Seznam CipherSuites , který lze použít ke komunikaci s produktem WebSphere MQ , lze najít v produktu “SSL CipherSpecs a CipherSuites v třídách WebSphere MQ pro prostředí Java” na stránce 683.

MQRC_SSL_CERTIFICATE_ODVOLÁNO
Certifikát prezentovaný správcem front byl nalezen v seznamu odvolaných certifikátů, který je zadán ve vlastnosti sslCertStores. Aktualizujte správce front tak, aby používal důvěryhodné certifikáty.

CHYBA MQRC_SSL_CERT_STORE_ERROR
U žádného z dodaných CertStores nebylo možné hledat certifikát předložený správcem front. Metoda MQException.getCause() vrací chybu, která se vyskytla při hledání prvního pokusu CertStore . Je-li příčinná výjimka NoSuchElementException, ClassCastException, nebo NullPointerException, zkontrolujte, že kolekce uvedená ve vlastnosti sslCertStores obsahuje alespoň jeden platný objekt CertStore .

SSL CipherSpecs a CipherSuites v třídách WebSphere MQ pro prostředí Java

Zda může aplikace produktu IBM WebSphere MQ classes for Java navázat připojení ke správci front, závisí na specifikaci CipherSpec na konci serveru MQI a na straně klienta CipherSuite určenou na straně klienta.

Pro každou kombinaci CipherSpec a CipherSuite, zda se aplikace IBM WebSphere MQ classes for Java může připojit ke správci front, závisí na hodnotě povinného pole sslFipsve třídě MQEnvironment nebo na hodnotě vlastnosti prostředí CMQC.SSL_FIPS_REQUIRED_PROPERTY.

Na konci kanálu MQI lze název CipherSpec zadat jako hodnotu parametru SSLCIPH u příkazu DEFINE CHANNEL CHLTYPE (SVRCONN). Na konci klienta kanálu MQI může aplikace IBM WebSphere MQ classes for Java nastavit pole sslCipherSuite ve třídě MQEnvironment nebo nastavit vlastnost prostředí CMQC.SSL_CIPHER_SUITE_PROPERTY.

Konfigurace vaší aplikace pro použití mapování IBM Java nebo Oracle Java CipherSuite

From IBM WebSphere MQ Version 7.5.0, opravná sada 5, you can configure whether your application uses the default IBM Java CipherSuite to WebSphere MQ CipherSpec mappings, or the Oracle CipherSuite to WebSphere MQ CipherSpec mappings. Proto můžete použít TLS CipherSuites , zda vaše aplikace používá prostředí JRE produktu IBM nebo prostředí Oracle JRE. Vlastnost systému Java `com.ibm.mq.cfg.useIBMCipherMappings` kontroluje, která mapování se používají. Vlastnost může mít jednu z následujících hodnot:

ano
Použijte mapování jazyka Java CipherSuite IBM na WebSphere MQ CipherSpec .

Tato hodnota je výchozí hodnotou.

ne

Použijte mapování Oracle CipherSuite na WebSphere MQ CipherSpec .

V následující tabulce jsou uvedeny seznamy CipherSpecs podporované produktem IBM WebSphere MQ a jejich ekvivalenty CipherSuites. Tabulka také určuje, zda může aplikace produktu IBM WebSphere MQ classes for Java navázat připojení ke správci front, je-li na konci serveru určena položka CipherSpec na straně serveru kanálu MQI a že je zadána hodnota CipherSuite na straně klienta.

Tabulka 89. CipherSpecs podporované produktem WebSphere MQ a jejich ekvivalenty CipherSuites

CipherSpec	**Ekvivalentní CipherSuite**	**Je možné připojení, pokud je SFIPS [1] nastavena na YES?**
NULL_MD5	SSL_RSA_WITH_NULL_MD5	Ne
NULL_SHA	SSL_RSA_WITH_NULL_SHA	Ne
RC4_MD5_EXPORT	SSL_RSA_EXPORT_WITH_RC4_40_MD5 (prostředíIBM JRE) Bez ekvivalentu pro prostředí Oracle JRE.	Ne
RC4_MD5_US	SSL_RSA_WITH_RC4_128_MD5	Ne
RC4_SHA_US	SSL_RSA_WITH_RC4_128_SHA (IBM JRE) Bez ekvivalentu pro prostředí Oracle JRE.	Ne
RC2_MD5_EXPORT	SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5 (IBM JRE) SSL_RSA_EXPORT_WITH_RC4_40_MD5 (Oracle JRE)	Ne
DES_SHA_EXPORT	SSL_RSA_WITH_DES_CBC_SHA (IBM JRE) Bez ekvivalentu pro prostředí Oracle JRE.	Ne
RC4_56_SHA_EXPORT1024	SSL_RSA_EXPORT1024_WITH_RC4_56_SHA (IBM JRE) Bez ekvivalentu pro prostředí Oracle JRE.	Ne
DES_SHA_EXPORT1024	SSL_RSA_EXPORT1024_WITH_DES_CBC_SHA (IBM JRE) Bez ekvivalentu pro prostředí Oracle JRE.	Ne
TRIPLE_DES_SHA_US	SSL_RSA_WITH_3DES_EDE_CBC_SHA (IBM JRE) Bez ekvivalentu pro prostředí Oracle JRE.	Ne
TLS_RSA_WITH_NULL_SHA256	SSL_RSA_WITH_NULL_SHA256 (prostředíIBM JRE) TLS_RSA_WITH_NULL_SHA256 (Oracle JRE)	Ne [7]
TLS_RSA_WITH_AES_128_CBC_SHA	SSL_RSA_WITH_AES_128_CBC_SHA (IBM JRE) TLS_RSA_WITH_AES_128_CBC_SHA (Oracle JRE)	Ano [5] [7]

Tabulka 89. CipherSpecs podporované produktem WebSphere MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	Ekvivalentní CipherSuite	Je možné připojení, pokud je SFIPS [1] nastavena na YES?
TLS_RSA_WITH_AES_128_CBC_SHA256	SSL_RSA_WITH_AES_128_CBC_SHA256 (IBM JRE) TLS_RSA_WITH_AES_128_CBC_SHA256 (Oracle JRE)	Ano [5] [7]
TLS_RSA_WITH_AES_256_CBC_SHA	SSL_RSA_WITH_AES_256_CBC_SHA (IBM JRE) TLS_RSA_WITH_AES_256_CBC_SHA (Oracle JRE)	Ano [5] [7]
TLS_RSA_WITH_AES_256_CBC_SHA256	SSL_RSA_WITH_AES_256_CBC_SHA256 (IBM JRE) TLS_RSA_WITH_AES_256_CBC_SHA256 (Oracle JRE)	Ano [5] [7]
AES_SHA_US[2]		
TLS_RSA_WITH_DES_CBC_SHA[8]	SSL_RSA_WITH_DES_CBC_SHA	Ne[3]
TLS_RSA_WITH_3DES_EDE_CBC_SHA[8] [9]	SSL_RSA_WITH_3DES_EDE_CBC_SHA	Ano
FIPS_WITH_DES_CBC_SHA	SSL_RSA_FIPS_WITH_DES_CBC_SHA (IBM JRE) Bez ekvivalentu pro prostředí Oracle JRE.	Ne[4]
FIPS_WITH_3DES_EDE_CBC_SHA	SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA (prostředíIBM JRE) Bez ekvivalentu pro prostředí Oracle JRE.	Ne[6]

Notes:

1. V aplikaci IBM WebSphere MQ classes for Java označte, že se mají použít pouze algoritmy certifikované podle standardu FIPS nastavením pole sslFipsRequired ve třídě MQEnvironment na `true` a označovat, že algoritmy non-FIPS lze použít také nastavením pole sslFipsRequired na `false`. Případně nastavte vlastnost prostředí CMQC.SSL_FIPS_REQUIRED_PROPERTY.
2. Tato CipherSpec nemá ekvivalent CipherSuite.
3. Tato CipherSpec byla certifikována FIPS 140-2 certifikovaná před 19th . květnem 2007.
4. Tato CipherSpec byla certifikována FIPS 140-2 certifikovaná před 19th . květnem 2007. Název FIPS_WITH_DES_CBC_SHA je historický a odráží fakt, že tato CipherSpec byla již dříve (ale již není) kompatibilní s FIPS-. Tato specifikace šifrování byla zamítnuta a její použití se nedoporučuje.
5. Tyto CipherSpecs (TLS_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_128_CBC_SHA256, TLS_RSA_WITH_AES_256_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA256) nelze použít k zabezpečení připojení z produktu WebSphere MQ Explorer ke správci front, pokud nejsou použity příslušné soubory neomezených zásad pro prostředí JRE používané průzkumníkem.

   Další informace o souborech zásad naleznete v tématu Informace o zabezpečení .
6. Název FIPS_WITH_3DES_EDE_CBC_SHA je historický a odráží fakt, že tato CipherSpec již byla (ale již není delší) kompatibilní s FIPS-. Tato specifikace šifrování byla zamítnuta a její použití se nedoporučuje.
7. Tyto CipherSpecs (TLS_RSA_WITH_NULL_SHA256, TLS_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA256, TLS_RSA_WITH_AES_256_CBC_SHA,

TLS_RSA_WITH_AES_256_CBC_SHA256) vyžadují prostředí IBM JRE 6.0 SR13 FP2 , 7.0 SR4 FP2 nebo novější.

8. Tyto CipherSpecs (TLS_RSA_WITH_3DES_EDE_CBC_SHA, TLS_RSA_WITH_DES_CBC_SHA, TLS_RSA_WITH_RC4_128_SHA256) mohou používat buď SSLv3 , nebo TLS. Je-li standard FIPS standardně povolen, použije se SSLv3 standardně. Chcete-li použít TLS, nastavte systémovou vlastnost Java **com.ibm.mq.cfg.preferTLS** na `true`.
9. Tato CipherSpec TLS_RSA_WITH_3DES_EDE_CBC_SHA byla zamítnuta. Nicméně lze ji přesto použít k přenosu až 32 GB dat před ukončením připojení s chybou AMQ9288. Chcete-li se této chybě vyhnout, je třeba při použití této CipherSpecbuď zabránit použití trojitého DES, nebo povolit resetování tajného klíče.

Související informace

Určení, že pro běhové prostředí klienta MQI je použit pouze certifikovaný standard FIPS CipherSpecs

Federální standardy zpracování informací (FIPS) pro UNIX, Linux a Windows

MQdev Blog: MQ Java, TLS Ciphers, non-IBM JRE & APARs IT06775, IV66840, IT09423, IT10837

MQdev blog: Vztah mezi produktem MQ CipherSpecs a šifry Java Cipher

Spuštění tříd produktu WebSphere MQ pro aplikace v jazyce Java

Pokud napíšete aplikaci (třída, která obsahuje metodu main ()) pomocí klienta nebo režimu vázání, spusťte program pomocí interpretu jazyka Java.

Použijte příkaz:

```
java -Djava.library.path=library_path MyClass
```

kde *cesta_knihovny* je cesta k třídám produktu WebSphere MQ pro knihovny Java (viz Třídy WebSphere MQ pro knihovny Java).

Třídy WebSphere MQ pro chování závislé na prostředí Java

Třídy WebSphere MQ for Java umožňují vytvářet aplikace, které mohou být spuštěny s různými verzemi produktu WebSphere MQ. Tato kolekce témat popisuje chování tříd Java závislých na těchto různých verzích.

Třídy WebSphere MQ pro jazyk Java poskytují jádro tříd, které zajišťují konzistentní funkci a chování ve všech prostředích. Funkce mimo toto jádro závisejí na schopnosti správce front, ke kterému je aplikace připojena.

Kromě zde uvedených informací se chování vykazuje způsobem popsaným v příručce Application Programming Reference, která je vhodná pro správce front.

Třídy jádra v třídách WebSphere MQ pro prostředí Java

Třídy WebSphere MQ pro jazyk Java obsahují hlavní sadu tříd, které lze použít ve všech prostředích.

Následující sada tříd je považována za základní třídy a lze ji použít ve všech prostředích pouze s menšími variantami uvedenými v seznamu “Omezení a variace pro třídy jádra produktu WebSphere MQ Classes for Java” na stránce 687.

- Prostředí MQEnvironment
- Výjimka MQException
- Volby MQGetMessage

  S výjimkou:

  – MatchOptions
  – GroupStatus
  – SegmentStatus

  - Segmentace
- MQManagedObject

  S výjimkou:
  - dotázat ()
  - nastavit ()
- Zpráva MQMessage

  S výjimkou:
  - groupId
  - messageFlags
  - messageSequenceČíslo
  - posunutí
  - originalLength
- MQPoolServices
- Událost MQPoolServices
- MQPoolServicesEventListener
- MQPoolToken
- Volby MQPutMessage

  S výjimkou:
  - KnownDestCount
  - UnknownDestCount
  - InvalidDestCount
  - recordFields
- Proces MQProcess
- MQQUEUE
- MQQueueManager

  S výjimkou:
  - začátek ()
  - accessDistributionList ()
- Správce MQSimpleConnectionManager
- MQTopic
- MQC

Poznámka:

1. Některé konstanty nejsou zahrnuty do jádra (podrobnosti viz “Omezení a variace pro třídy jádra produktu WebSphere MQ Classes for Java” na stránce 687); nepoužívat je v kompletních přenosných programech.
2. Některé platformy nepodporují všechny režimy připojení. Na těchto platformách můžete používat pouze základní třídy a volby, které souvisejí s podporovanými režimy. (Viz “Volby připojení pro třídy WebSphere MQ pro prostředí Java” na stránce 632.)

Omezení a variace pro třídy jádra produktu WebSphere MQ Classes for Java

Hlavní třídy se obecně chovají konzistentně ve všech prostředích, a to i v případě, že ekvivalentní volání MQI normálně mají rozdíly prostředí. The behavior is as if a Windows, UNIX or Linux WebSphere MQ queue manager is used, except for the following minor restrictions and variations.

*Omezení pro hodnoty MQGMO_ * v třídách WebSphere MQ pro prostředí Java*
Určité hodnoty MQGMO_ * nejsou podporovány všemi správci front.

Při použití následujících hodnot MQGMO_ * může dojít k vyvolání výjimky MQException z objektu MQQueue.get():

MQGMO_SYNCPOINT_IF_PERSISTENT
MQGMO_MARK_ SKIP_BACKOUT
MQGMO_BROWSE_MSG_UNDER_CURSOR
MQGMOVÝ_ZÁMEK
MQGMO_ODEMKNOUT
MQGMO_LOGICAL_ORDER
ZPRÁVA MQGMO_COMPLETE_MESSAGE
MQGMO_ALL_MSGS_AVAILABLE
DOSTUPNÉ MQGMO_ALL_SEGMENTS_AVAILABLE
MQGMO_UNMARKED_BROWSE_MSG,
POPISOVAČ MQGMO_MARK_ BROWSE_HANDLE
MQGMO_MARKER_BROWSE_CO_OP
POPISOVAČ MQGMO_UNMARK_ BROWSE_HANDLE
MQGMO_UNMARK_ BROWSE_CO_OP

Kromě toho není funkce MQGMO_SET_SIGNAL v případě použití jazyka Java podporována.

*Omezení pro hodnoty MQPMRF_ * v třídách WebSphere MQ pro prostředí Java*
Ty se používají pouze při vkládání zpráv do distribučního seznamu a jsou podporovány pouze správci front podporujícím distribuční seznamy. Například, správci front z/OS nepodporují distribuční seznamy.

*Omezení pro hodnoty MQPMO_ * v třídách WebSphere MQ pro prostředí Java*
Některé hodnoty MQPMO_ * nejsou podporovány všemi správci front

Použití následujících hodnot MQPMO_ * může vést k vyvolání výjimky MQException z objektu MQQueue.put() nebo objektu MQQueueManager.put ():

MQPMO_LOGICAL_ORDER
MQPMO_NOVÉ_ KOREL_ID
MQPMO_NOVÉ_ID_ZPRÁVY
MQPMOD_RESOLVE_LOKÁLNÍ_Q

*Omezení a variace pro hodnoty MQCNO_ * ve třídách WebSphere MQ pro prostředí Java*
Určité hodnoty `MQCNO_*` nejsou podporovány.

- Automatické opětovné připojení klienta není podporováno třídami WebSphere MQ pro jazyk Java. Bez ohledu na hodnotu `MQCNO_RECONNECT_*` , kterou jste nastavili, se bude nadále chovat jako, jako byste nastavili `MQCNO_RECONNECT_DISABLED`.
- Produkt `MQCNO_FASTPATH` je ignorován ve správcích front, které nepodporují produkt `MQCNO_FASTPATH`. Je také ignorován klientskými připojeními.

*Omezení pro hodnoty MQRO_ * ve třídách WebSphere MQ pro prostředí Java*
Mohou být nastaveny následující volby sestavy.

MQRO_EXCEPTION_WITH_FULL_DATA
MQRO_EXPIRATION_WITH_FULL_DATA
MQRO_COA_WITH_FULL_DATA
MQRO_COD_WITH_FULL_DATA
MQRO_DISCARD_MSG
MQRO_PASS_DISCARD_AND_EXPIRY

Další informace viz Sestava.

Funkce mimo hlavní třídy tříd produktu WebSphere MQ pro prostředí Java

Třídy WebSphere MQ pro prostředí Java obsahují určité funkce, které jsou speciálně navrženy pro použití rozšíření rozhraní API, která nejsou podporována všemi správci front. Tato kolekce témat popisuje, jak se chovají při použití správce front, který je nepodporuje (*nepodporuje*).

Variace ve volbě konstruktoru MQQueueManager

Některé z konstruktorů MQQueueManager obsahují volitelný celočíselný argument. Některé hodnoty tohoto argumentu nejsou akceptovány na všech platformách.

Pokud konstruktor MQQueueManager obsahuje volitelný celočíselný argument, mapuje se do pole voleb MQCNO rozhraní MQI a používá se k přepínání mezi normálním a rychlým připojením cesty. Tato přídavná forma konstruktoru je přijata ve všech prostředích, pokud jsou jediné použité volby MQCNO_STANDARD_BINDING nebo MQCNO_FASTPATH_BINDING. Všechny ostatní volby způsobí selhání konstruktoru při chybě MQRC_OPTIONS_ERROR. Volba rychlé cesty CMQC.MQCNO_FASTPATH_BINDING je dodržován pouze s vazbami připojení ke správci front, který ji podporuje. V jiných prostředích je ignorována.

Omezení pro metodu .begin () produktu MQQueueManager

Tuto metodu lze použít pouze pro správce front produktu WebSphere MQ v systémech UNIX, Linuxnebo Windows v režimu vazeb. Jinak dojde k selhání funkce MQRC_ENVIRONMENT_ERROR.

Další informace viz část “Koordinace JTA/JDBC používající třídy WebSphere MQ pro prostředí Java” na stránce 676.

Variace v polích Volby MQGetMessage

Někteří správci front nepodporují strukturu MQGMO verze 2, takže je třeba nastavit některá pole na jejich výchozí hodnoty.

Používáte-li správce front, který nepodporuje strukturu MQGMO verze 2, ponechejte následující pole nastavovaná na výchozí hodnoty:

GroupStatus
SegmentStatus
Segmentace

Také pole MatchOptions podporuje pouze MQMO_MATCH_MSG_ID a MQMO_MATCH_CORREL_ID. Pokud do těchto polí zadáte nepodporované hodnoty, následující MQDestination.get() selže s chybou MQRC_GMO_ERROR. Pokud správce front nepodporuje strukturu MQGMO verze 2, tato pole se neaktualizují po úspěšném provedení operace MQDestination.get().

Omezení v distribučních seznamech ve třídách WebSphere MQ pro prostředí Java

Ne všichni správci front umožňují otevřít objekt MQDistributionList.

K vytváření distribučních seznamů se používají následující třídy:

MQDistributionList
Položka MQDistributionList
MQMessageTracker

Můžete vytvořit a naplnit položky MQDistributionLists a MQDistributionListv libovolném prostředí, ale ne všechny správce front umožňují otevřít MQDistributionList. Správci front z/OS zejména nepodporují distribuční seznamy. Pokus o otevření objektu MQDistributionList při použití takového správce front vede k výjimce MQRC_OD_ERROR.

Variace v polích Volby MQPutMessage

Pokud správce front nepodporuje distribuční seznamy, budou s některými poli MQPMO zacházeno jinak.

Čtyři pole v objektu MQPMO jsou vykreslena jako následující členské proměnné ve třídě Volby MQPutMessage:

KnownDestCount
UnknownDestCount

InvalidDestCount
recordFields

Tato pole jsou primárně určena pro použití s rozdělovníky. Avšak správce front, který podporuje distribuční seznamy, vyplní také pole DestCount po provedení MQPUT do jediné fronty. Například, pokud je fronta vyřešena na lokální frontu, knownDestPočet je nastaven na 1 a ostatní dvě pole počtu jsou nastavena na 0.

Pokud správce front nepodporuje distribuční seznamy, jsou tyto hodnoty simulovány následujícím způsobem:

- Je-li funkce put () úspěšná, unknownDestCount je nastaven na 1 a ostatní jsou nastaveny na 0.
- Pokud funkce put () selže, invalidDestPočet je nastaven na hodnotu 1 a ostatní jsou nastaveny na 0.

Proměnná recordFields se používá s rozdělovníky. Hodnota může být kdykoli zapsána do recordFields , bez ohledu na prostředí. Pokud je objekt Volby MQPutMessagepoužit v následném objektu MQDestination.put() nebo MQQueueManager.put () a nikoli MQDistributionList.put (), je tento parametr ignorován.

Omezení v polích MQMD s třídami WebSphere MQ pro prostředí Java
Při použití správce front, který nepodporuje segmentaci, by se při použití správce front, která nepodporuje segmentaci, měla při výchozím nastavení ponechat určitá pole MQMD se segmentací zpráv.

Následující pole MQMD jsou z velké části znepokojují segmentací zpráv:

GroupId
MsgSeqNumber
Offset
MsgFlags
OriginalLength

Pokud aplikace nastaví jakékoli z těchto polí MQMD na jiné hodnoty než jejich výchozí hodnoty a pak funkci put () nebo get () ve správci front, který tyto hodnoty nepodporuje, vrátí funkce put () nebo get () výjimku MQException s MQRC_MD_ERROR. Úspěšná funkce put () nebo get () s takovým správcem front vždy ponechá pole MQMD nastavenou na jejich výchozí hodnoty. Neposílejte seskupenou nebo segmentovanou zprávu do aplikace v jazyce Java, která je spuštěna proti správci front, který nepodporuje seskupování zpráv a segmentaci.

Pokusí-li se aplikace jazyka Java 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), je načtena bez chyby. Nicméně pole MQMD ve zprávě MQMessage nejsou aktualizována, vlastnost formátu MQMessage je nastavena na hodnotu MQFMT_MD_EXTENSION a před daty ve struktuře MQMDE, která obsahuje hodnoty pro nová pole, bude použita předpona MQMMT_MD_EXTENSION.

Omezení pro třídy WebSphere MQ pro prostředí Java pod serverem CICS Transaction Server

V prostředí CICS Transaction Server pro prostředí z/OS je povoleno vydávat volání CICS nebo WebSphere MQ pouze pro hlavní (první) podproces.

Všimněte si, že třídy JMS WebSphere MQ nejsou podporovány pro použití v aplikaci CICS Java.

Z tohoto důvodu není možné sdílet objekty MQQueueManager nebo MQQueue mezi podprocesy v tomto prostředí nebo vytvořit nový objekt MQQueueManager v podřízeném podprocesu.

Spuštění tříd produktu IBM WebSphere MQ pro aplikace Java v rámci platformy Java Enterprise Edition

Existují určitá omezení a aspekty návrhu, které musí být vzaty v úvahu před použitím tříd produktu IBM WebSphere MQ pro produkt Java v prostředí Java EE

Třídy IBM WebSphere MQ pro produkt Java mají omezení při použití v prostředí Java EE . Při navrhování, implementaci a správě tříd produktu IBM WebSphere MQ pro aplikaci Java , které jsou spuštěny

v prostředí Java EE , je třeba vzít v úvahu také další aspekty, které je třeba vzít v úvahu. Tato omezení a pokyny jsou popsány v následujících sekcích.

Omezení transakcí JTA

Jediný podporovaný správce transakcí pro aplikace používající třídy IBM WebSphere MQ pro produkt Java je sám o sobě IBM WebSphere MQ . Přestože aplikace pod kontrolou JTA může používat třídy IBM WebSphere MQ pro produkt Java, žádná práce prováděná prostřednictvím těchto tříd není řízena jednotkami JTA. Namísto toho tvoří lokální jednotky práce oddělené od těch, které jsou spravovány aplikačním serverem prostřednictvím rozhraní JTA. Zejména žádné odvolání transakce JTA nevedlo k odvolání žádných odeslaných nebo přijatých zpráv. Toto omezení platí pro aplikace spravované transakce nebo pro transakce spravované kontejnerem a pro všechny kontejnery spravované kontejnerem Java EE . Chcete-li provádět práci systému zpráv přímo s produktem IBM WebSphere MQ uvnitř transakcí koordinovaných aplikačním serverem, musí být místo toho použity třídy IBM WebSphere MQ pro platformu JMS.

Vytvoření podprocesů

Třídy IBM WebSphere MQ pro produkt Java vytváří podprocesy vnitřně pro různé operace. Například při spuštění v režimu BINDINGS pro volání přímo v lokálním správci front jsou volání prováděna v podprocesu worker vytvořeném interně třídami produktu IBM WebSphere MQ pro produkt Java. Další podprocesy lze vytvořit interně, například vymazat nepoužívaná připojení z fondu připojení nebo odebrat odběry pro ukončené aplikace typu publikování/odběr.

Některé aplikace Java EE (například ty, které jsou spuštěny v kontejnerech EJB a webových kontejnerů), nesmějí vytvářet nové podprocesy. Místo toho se musí všechny práce provádět na hlavních aplikačních podprocesech spravovaných aplikačním serverem. Když aplikace používají třídy IBM WebSphere MQ pro produkt Java, nemusí být aplikační server schopen rozlišit mezi kódem aplikace a třídami IBM WebSphere MQ pro kód Java , takže dříve popsané podprocesy způsobí, že aplikace bude nevyhovující specifikaci kontejneru. Třídy produktu IBM WebSphere MQ pro platformu JMS nepřeruší tyto specifikace prostředí Java EE a lze je tedy použít.

Bezpečnostní omezení

Zásady zabezpečení implementované aplikačním serverem mohou zabránit určitým operacím, které jsou prováděny třídami IBM WebSphere MQ pro rozhraní API produktu Java , jako je například vytváření a provoz nových podprocesů ovládacího prvku (jak je popsáno v předchozích sekcích).

Aplikační servery jsou například standardně spuštěny se zabezpečením produktu Java a umožňují jeho povolení prostřednictvím některé konfigurace specifické pro aplikační server (některé aplikační servery také umožňují podrobnější konfiguraci zásad používaných v produktu Java Security). Je-li aktivována služba zabezpečení produktu Java , třídy produktu IBM WebSphere MQ pro produkt Java mohou porušit pravidla podprocesů pro zásady zabezpečení produktu Java definovaná pro aplikační server a rozhraní API nemusí být schopno vytvořit všechny podprocesy, které potřebuje ke své funkci. Chcete-li zabránit problémům se správou podprocesů, použití tříd produktu IBM WebSphere MQ pro produkt Java není podporováno v prostředích, ve kterých je povoleno zabezpečení produktu Java .

Aspekty izolace aplikace

Zamýšleným přínosem pro spouštění aplikací v prostředí Java EE je izolace aplikace. Návrh a implementace tříd produktu IBM WebSphere MQ pro produkt Java předdatum prostředí Java EE . IBM WebSphere MQ třídy pro Java lze použít způsobem, který nepodporuje koncepci izolace aplikace. Konkrétní příklady aspektů v této oblasti zahrnují:

- Použití statického nastavení (prostředí JVM) v rámci třídy MQEnvironment, například:
  - ID uživatele a heslo, které má být použito pro identifikaci a ověření připojení
  - název hostitele, port a kanál použité pro připojení klienta
  - Konfigurace zabezpečení SSL pro zabezpečená připojení klienta

Úprava kterékoli z vlastností MQEnvironment ve prospěch jedné aplikace ovlivní také jiné aplikace používající stejné vlastnosti. Při spuštění v prostředí s více aplikacemi, jako např. Java EE, musí každá aplikace používat svou vlastní odlišnou konfiguraci prostřednictvím vytváření objektů MQQueueManager se specifickou sadou vlastností, nikoli jako výchozí nastavení vlastností konfigurovaných v rámci třídy MQEnvironment pro celý proces.

- Třída MQEnvironment zavádí řadu statických metod, které se chovají globálně na všech aplikacích používajících třídy IBM WebSphere MQ pro Java v rámci stejného procesu JVM, a neexistuje žádný způsob, jak potlačit toto chování pro určité aplikace. Příklady:
  - konfigurace vlastností SSL, jako je umístění úložiště klíčů
  - konfigurace uživatelských procedur kanálu klienta
  - povolení nebo zakázání trasování diagnostiky
  - správa výchozího fondu připojení používaného k optimalizaci využití připojení ke správcům front

  Vyvolání těchto metod ovlivní všechny aplikace spuštěné ve stejném prostředí Java EE .
- Sdružování připojení je povoleno, aby byl optimalizován proces vytváření více připojení ke stejnému správci front. Výchozí správce fondu připojení je celoprocesový a je sdílen více aplikacemi. Změny v konfiguraci fondu připojení, jako je například nahrazení výchozího správce připojení pro jednu aplikaci pomocí metody MQEnvironment.setDefaultConnectionManager() proto ovlivňují jiné aplikace spuštěné ve stejném aplikačním serveru Java EE .
- Zabezpečení SSL je konfigurováno pro aplikace používající třídy IBM WebSphere MQ pro produkt Java pomocí vlastností třídy MQEnvironment a vlastností objektu MQQueueManager . Není integrován se spravovanou konfigurací zabezpečení samotného aplikačního serveru. Musíte se ujistit, že jste nakonfigurovali třídy IBM WebSphere MQ pro produkt Java odpovídajícím způsobem, abyste poskytli požadovanou úroveň zabezpečení a nepoužívali konfiguraci aplikačního serveru.

Omezení režimu vazeb

Produkt IBM WebSphere MQ a WebSphere Application Server lze instalovat na stejném počítači, jako jsou hlavní verze správce front a adaptéru prostředků produktu IBM WebSphere MQ (RA) dodávané v produktu WebSphere Application Server. Například produkt WebSphere Application Server Version 7.0, který se dodává s úrovní IBM WebSphere MQ RA produktu 7.0.1, může být nainstalován na stejném počítači jako správce front produktu Version 6.0 .

Pokud se správce front a hlavní verze adaptéru prostředků liší, nelze použít připojení vazeb. Veškerá připojení ze serveru WebSphere Application Server ke správci front s použitím adaptéru prostředků musí používat připojení typu klienta. Spojení vazeb lze použít, pokud jsou verze stejné.

Použití tříd produktu WebSphere MQ pro službu JMS

Třídy WebSphere MQ pro platformy JMS (WebSphere MQ Classes for JMS) jsou poskytovatelem rozhraní JMS dodávaným s produktem WebSphere MQ. Kromě implementace rozhraní definovaných v balíku javax.jms produkt WebSphere MQ Classes for JMS poskytuje dvě sady rozšíření rozhraní JMS API.

Specifikace JMS definuje sadu rozhraní, která mohou aplikace používat k provádění operací systému zpráv. Balík javax.jms definuje rozhraní JMS a poskytovatel platformy JMS implementuje tato rozhraní pro specifický produkt systému zpráv. Produkt WebSphere MQ verze 7.5 aktuálně používá specifikaci JMS 1.1 . Třídy WebSphere MQ pro systém JMS je poskytovatelem rozhraní JMS, který implementuje rozhraní JMS pro produkt WebSphere MQ.

Specifikace JMS očekává objekty ConnectionFactory a Destination s objekty, které mají být spravovány. Administrátor vytváří a udržuje spravované objekty v centrálním úložišti a aplikace JMS načte tyto objekty pomocí rozhraní JNDI (Java Naming and Directory Interface). Třídy WebSphere MQ pro platformu JMS podporují použití spravovaných objektů a administrátor může pomocí nástroje pro administraci produktu WebSphere MQ JMS nebo Průzkumníka produktu WebSphere MQ vytvářet a spravovat spravované objekty.

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

Rozšíření WebSphere MQ JMS
Předchozí vydání tříd WebSphere MQ pro rozhraní JMS obsahují rozšíření, která jsou implementována v objektech, jako jsou objekty MQConnectionFactory, MQQueue a MQTopic. Tyto objekty mají vlastnosti a metody, které jsou specifické pro produkt WebSphere MQ. Objekty mohou být spravovanými objekty nebo aplikace může dynamicky vytvářet objekty za běhu. Toto vydání tříd produktu WebSphere MQ pro platformu JMS udržuje tato rozšíření, která jsou nyní známá jako rozšíření produktu WebSphere MQ JMS. Můžete pokračovat v používání, beze změny, jakýchkoli aplikací, které používají tato rozšíření.

Rozšíření IBM JMS
Toto vydání tříd produktu WebSphere MQ pro službu JMS poskytuje generičtější sadu rozšíření rozhraní JMS API, která nejsou specifická pro produkt WebSphere MQ jako systém zasílání zpráv. Tato rozšíření jsou známá jako rozšíření IBM JMS a mají následující obecné cíle:

- Chcete-li zajistit vyšší úroveň konzistence mezi poskytovateli platformy JMS IBM ,
- Chcete-li usnadnit zápis do aplikace mostu mezi dvěma systémy zasílání zpráv IBM
- Chcete-li usnadnit portům aplikaci z jednoho poskytovatele platformy JMS IBM do jiného

Rozšíření poskytují funkce podobné funkcím, které jsou k dispozici v Message Service Client for C/C++ a v Message Service Client for .NET.

Proč bych měl používat třídy WebSphere MQ pro platformu JMS?

Použití tříd produktu WebSphere MQ pro platformu JMS má následující výhody:

- Schopnosti platformy JMS můžete znovu použít.

  Třídy WebSphere MQ pro rozhraní JMS je poskytovatelem JMS, který implementuje rozhraní JMS pro systém WebSphere MQ jako systém zpráv. Je-li vaše organizace nová pro produkt WebSphere MQ, ale již má zkušenosti s vývojem aplikací JMS, může být snazší použít známé rozhraní API JMS pro přístup k prostředkům produktu WebSphere MQ místo jednoho z dalších rozhraní API poskytovaných s produktem WebSphere MQ.
- Platforma JMS je integrální součástí prostředí Java Platform, Enterprise Edition (Java EE).

  Rozhraní JMS je přirozeným rozhraním API, které má být používáno pro systém zpráv na platformě Java EE . Každý aplikační server, který je kompatibilní s prostředím Java EE , musí obsahovat poskytovatele rozhraní JMS. Můžete použít rozhraní JMS v aplikačních klientech, servletech, stránkách JavaServer (JSP), objektech EJB (Enterprise Java Bean) a objektů 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.
- Administrátor může vytvářet a spravovat spravované objekty platformy JMS v centrálním úložišti a třídy produktu WebSphere MQ pro aplikace JMS mohou tyto objekty načítat pomocí rozhraní JNDI (Java Naming and Directory Interface).

  Továrny připojení JMS a cíle zapouzdřují specifické informace o produktu WebSphere MQ , jako jsou například názvy správců front, názvy kanálů, volby připojení, názvy front a názvy témat. Pokud jsou továrny připojení a místa určení uloženy jako spravované objekty, není tato informace pevně zakódována do aplikace. Toto uspořádání proto poskytuje aplikaci se stupněm nezávislosti na základní konfiguraci produktu WebSphere MQ .
- Služba JMS je odvětvovým standardním rozhraním API, které může poskytovat přenositelnost aplikací.

  Aplikace platformy JMS může prostřednictvím rozhraní JNDI načítat továrny připojení a cíle, které jsou uloženy jako spravované objekty, a používat pouze rozhraní definovaná v balíku javax.jms k provádění operací systému zpráv. Aplikace je pak zcela nezávislá na všech poskytovatelích JMS, jako jsou například třídy WebSphere MQ pro JMS a může být přenesena z jednoho poskytovatele rozhraní JMS do jiného bez jakýchkoli změn v aplikaci.

Pokud rozhraní JNDI není k dispozici v konkrétním prostředí aplikace, mohou třídy WebSphere MQ pro aplikaci JMS používat rozšíření rozhraní JMS API k dynamickému vytváření a konfiguraci továren připojení a cílů v běhovém prostředí. Aplikace je poté zcela samostatná, ale je vázána na třídy WebSphere MQ pro JMS jako poskytovatele rozhraní JMS.

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

  Aplikace mostu je aplikace, která přijímá zprávy z jednoho systému zpráv a odesílá je do jiného systému zasílání zpráv. Zápis aplikace mostu může být komplikovaný pomocí rozhraní API a formátů zpráv specifických pro produkt. Namísto toho můžete napsat aplikaci pro přemostění pomocí dvou poskytovatelů JMS, jeden pro každý systém zasílání zpráv. Aplikace pak použije pouze jedno rozhraní API, rozhraní JMS API a zpracovává pouze zprávy JMS.

Začínáme s třídami produktu WebSphere MQ pro službu JMS

Toto téma poskytuje přehled tříd produktu WebSphere MQ pro službu JMS a informuje o tom, co potřebujete vědět před použitím tříd produktu WebSphere MQ pro službu JMS.

Nezbytné předpoklady pro třídy WebSphere MQ pro JMS

Chcete-li vyvinout a spustit třídy WebSphere MQ pro aplikace JMS, potřebujete určité softwarové komponenty jako nezbytné předpoklady.

Nejnovější informace o předpokladech pro třídy produktu WebSphere MQ pro službu JMS naleznete v souboru Readme produktu WebSphere MQ .

Chcete-li vyvíjet třídy WebSphere MQ pro aplikace JMS, potřebujete sadu Java 2 Software Development Kit (SDK). Podrobnosti o sad JDK podporovaných vaším operačním systémem naleznete na stránce systémových požadavků produktu WebSphere MQ . Viz WebSphere MQ Requirements.

Chcete-li spustit třídy WebSphere MQ pro aplikace JMS, potřebujete tyto softwarové komponenty:

- Správce front produktu WebSphere MQ
- Běhové prostředí Java Runtime Environment (JRE) pro každý systém, na kterém spouštíte aplikace.

Požadujete-li připojení SSL k použití šifrovacích modulů, které mají certifikaci FIPS 140-2, potřebujete poskytovatele FIPS IBM Java JSSE (IBMJSSEFIPS). Každá sada IBM Java 2 SDK a JRE ve verzi 5 nebo novější obsahuje IBMJSSEFIPS.

Můžete použít adresy Internet Protocol verze 6 (IPv6) ve vašich třídách WebSphere MQ pro aplikace JMS za předpokladu, že jsou adresy IPv6 podporovány vaším prostředím JVM (Java Virtual Machine) a implementací TCP/IP ve vašem operačním systému. Administrační nástroj WebSphere MQ JMS (viz “Použití nástroje pro administraci produktu WebSphere MQ JMS” na stránce 902) také přijímá adresy IPv6 .

Nástroj pro administraci produktu WebSphere MQ JMS a produkt WebSphere MQ Explorer používají rozhraní JNDI (Java Naming and Directory Interface) pro přístup k adresářové službě, která ukládá spravované objekty. Třídy WebSphere MQ pro aplikace JMS mohou také používat rozhraní JNDI k načítání spravovaných objektů z adresářové služby. Poskytovatel služeb je kód, který poskytuje přístup k adresářové službě mapováním volání JNDI k volání na adresářovou službu. S třídami produktu WebSphere MQ pro službu JMS jsou dodávány následující poskytovatelé služeb:

- Poskytovatel služby LDAP (Lightweight Directory Access Protocol) v souborech ldap.jar a providerutil.jar. Poskytovatel služeb LDAP poskytuje přístup k adresářové službě založené na serveru LDAP.
- Poskytovatel služeb souborového systému v souborech fscontext.jar a providerutil.jar. Poskytovatel služeb systému souborů poskytuje přístup k adresářové službě založené na lokálním systému souborů.

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

Příprava programů JMS pro klienta IBM WebSphere MQ pro produkt HP Integrity NonStop Server

Toto téma vysvětluje, co potřebujete vědět, než začnete vyvíjet a spouštět programy platformy JMS pro klienta IBM WebSphere MQ pro produkt HP Integrity NonStop Server.

Třídy IBM WebSphere MQ pro platformu JMS jsou instalovány jako součást instalace klienta IBM WebSphere MQ pro instalaci produktu HP Integrity NonStop Server . Podrobnosti o souhrnu obsahu instalace najdete v tématu Systém souborů.

Některé aspekty funkčnosti klienta jsou specifické pro hostitelský operační systém. Další informace o podporovaných funkcích klienta produktu IBM WebSphere MQ pro produkt HP Integrity NonStop Servernaleznete v tématu KlientIBM WebSphere MQ pro podporovaná prostředí a funkce produktu HP Integrity NonStop Server.

Požadavky

Chcete-li sestavit a spustit aplikace JMS, musí být nainstalována komponenta *HP Integrity NonStop Server for Java* a dostupná.

Nastavení

Informace o nastavení prostředí pro spouštění a sestavení aplikací, ve kterých můžete použít třídy produktu IBM WebSphere MQ pro platformu JMS, najdete v tématu “Proměnné prostředí používané třídami produktu IBM WebSphere MQ pro platformu JMS” na stránce 699.

Informace o krocích nezbytných ke konfiguraci správce front pro příjem žádostí o připojení z klientských aplikací viz “Poinstalační nastavení pro třídy produktu WebSphere MQ pro aplikace JMS” na stránce 746.

Další informace o ověření platnosti tříd produktu IBM WebSphere MQ pro prostředí JMS viz “Ověřovací test dvoubodové instalace pro třídy WebSphere MQ pro JMS” na stránce 749.

Zapisování aplikací

Další informace o zápisu aplikací JMS naleznete v tématu “Zápis tříd produktu WebSphere MQ pro aplikace JMS” na stránce 778.

Další informace o použití nástroje pro administraci produktu IBM WebSphere MQ JMS naleznete v příručce “Použití nástroje pro administraci produktu WebSphere MQ JMS” na stránce 902.

Ukázky

Ukázkové aplikace jsou k dispozici v následujícím podadresáři instalace: `opt/mqm/samp/jms`.

Další informace o krocích konfigurace nezbytných ke spuštění ukázek viz “Příprava a spuštění ukázkových programů” na stránce 104.

Řešení problémů

Informace o řešení problémů najdete v tématu “Řešení problémů s třídami produktu IBM WebSphere MQ pro platformu JMS” na stránce 769.

Instalace a konfigurace tříd produktu WebSphere MQ pro službu JMS

Tato sekce popisuje adresáře a soubory, které jsou vytvořeny při instalaci tříd WebSphere MQ pro službu JMS a dozvíte se, jak nakonfigurovat třídy produktu WebSphere MQ pro platformu JMS po instalaci.

Související pojmy

“Co je nainstalováno pro třídy IBM WebSphere MQ pro JMS” na stránce 697

Při instalaci tříd produktu IBM WebSphere MQ pro platformu JMS se vytvoří počet souborů a adresářů. V systému Windows se některá konfigurace provádí při instalaci automatickým nastavením proměnných

prostředí. Na jiných platformách a v určitých prostředích Windows musíte nastavit proměnné prostředí, než budete moci spouštět třídy IBM WebSphere MQ pro aplikace JMS.

“Spuštění tříd produktu WebSphere MQ pro aplikace JMS v rámci správce zabezpečení produktu Java” na stránce 706
Třídy WebSphere MQ pro platformu JMS lze spustit s povoleným správcem zabezpečení produktu Java . Chcete-li úspěšně spustit aplikace s povoleným správcem zabezpečení, musíte nakonfigurovat prostředí JVM (Java virtual machine) s vhodným konfiguračním souborem zásad.

“Adaptér prostředků produktu IBM WebSphere MQ” na stránce 710
Adaptér prostředků umožňuje aplikacím spuštěným na aplikačním serveru přístup k prostředkům produktu IBM WebSphere MQ . Podporuje příchozí a odchozí komunikaci.

“Poinstalační nastavení pro třídy produktu WebSphere MQ pro aplikace JMS” na stránce 746
Toto téma informuje o tom, jaké oprávnění potřebují třídy WebSphere MQ pro aplikace JMS, aby bylo možné získat přístup k prostředkům správce front. Také uvádí režimy připojení a popisuje, jak nakonfigurovat správce front, aby se aplikace mohly připojit v režimu klienta.

“Ověřovací test dvoubodové instalace pro třídy WebSphere MQ pro JMS” na stránce 749
Program IVT (point-to-point installation verification test) je dodáván s třídami WebSphere MQ pro platformu JMS. 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.

“Ověřovací test instalace publikování/odběru pro třídy WebSphere MQ pro JMS” na stránce 752
Program IVT (publish/subscribe installation verification test) je dodáván s třídami WebSphere MQ pro platformu 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.

“Testovací program ověření instalace pro adaptér prostředků WebSphere MQ” na stránce 756
Program IVT je dodáván jako soubor EAR. Chcete-li použít tento program, musíte jej implementovat a definovat některé objekty jako prostředky JCA.

“Konfigurace adaptéru prostředků pro odchozí komunikaci” na stránce 727
Chcete-li konfigurovat odchozí komunikaci, definujte vlastnosti objektu ConnectionFactory a spravovaného cílového objektu.

“Podpora pro OSGi” na stránce 768
OSGi poskytuje rámec, který podporuje implementaci aplikací jako balíky. Devět svazků balíků OSGi je dodáváno jako součást produktu IBM WebSphere MQ classes for JMS .

“Řešení problémů s třídami produktu IBM WebSphere MQ pro platformu JMS” na stránce 769
Můžete vyšetřit problémy spuštěním programů pro ověření instalace a pomocí zařízení pro trasování a protokolování.

Související úlohy

“Instalace a testování adaptéru prostředků MQ v produktu WAS CE” na stránce 758
Instalace adaptéru prostředků IBM WebSphere MQ a spuštění aplikace testu verifikace instalace (IVT) v produktu WebSphere Application Server CE.

“Implementace aplikace IVT na WAS CE s vlastním prostředím MQ” na stránce 760
Chcete-li použít jinou frontu, správce front, port, hostitele, kanál nebo použít režim vazeb namísto režimu klienta, je třeba před implementací adaptéru prostředků nebo aplikace IVT upravit aplikaci IVT a přidružené skripty v produktu WebSphere Application Server CE.

“Implementace aplikace IVT v produktu JBoss s vlastním prostředím produktu IBM WebSphere MQ” na stránce 763
Při instalaci adaptéru prostředků produktu IBM WebSphere MQ v produktu JBoss, pokud chcete použít jinou frontu, správce front, port, hostitele, kanál nebo použít režim vazeb namísto režimu klienta, je třeba nejprve upravit aplikaci IVT a přidružené skripty v produktu JBoss před implementací adaptéru prostředků nebo aplikace IVT.

Určování problémů pro adaptér prostředků produktu IBM WebSphere MQ

Související odkazy

“Skripty poskytnuté s třídami produktu WebSphere MQ pro službu JMS” na stránce 767
Je k dispozici řada skriptů, které pomáhají s běžnými úlohami, které je třeba provést při použití tříd produktu WebSphere MQ pro službu JMS.

Co je nainstalováno pro třídy IBM WebSphere MQ pro JMS

Při instalaci tříd produktu IBM WebSphere MQ pro platformu JMS se vytvoří počet souborů a adresářů. V systému Windows se některá konfigurace provádí při instalaci automatickým nastavením proměnných prostředí. Na jiných platformách a v určitých prostředích Windows musíte nastavit proměnné prostředí, než budete moci spouštět třídy IBM WebSphere MQ pro aplikace JMS.

Pro většinu operačních systémů jsou třídy produktu IBM WebSphere MQ pro platformu JMS nainstalovány jako volitelná komponenta při instalaci produktu IBM WebSphere MQ. Pro klienta produktu IBM WebSphere MQ pro produkt HP Integrity NonStop Serverjsou standardně nainstalovány třídy produktu IBM WebSphere MQ pro platformu JMS. Další informace o instalaci produktu IBM WebSphere MQnaleznete v následujících tématech:

Instalace serveru WebSphere MQ
Instalace klienta IBM WebSphere MQ

Tabulka 90 na stránce 697 zobrazuje, kde jsou na každé platformě instalovány třídy IBM WebSphere MQ pro soubory JMS.

Tabulka 90. Třídy IBM WebSphere MQ pro instalační adresáře platformy JMS

Platforma	**Adresář**
AIX	*MQ_INSTALLATION_PATH*/java.
HP Integrity NonStop Server	*MQ_INSTALLATION_PATH*/opt/mqm/java
HP-UX, Linuxa Solaris	*MQ_INSTALLATION_PATH*/java.
Windows	*MQ_INSTALLATION_PATH*\java
MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM WebSphere MQ .	

Instalační adresář zahrnuje:

- Třídy IBM WebSphere MQ pro soubory JAR platformy JMS, které jsou umístěny v adresáři *MQ_INSTALLATION_PATH*\java\lib.
- Nativní knihovny produktu IBM WebSphere MQ , které jsou používány aplikacemi, které používají nativní rozhraní Java.

  32bitové nativní knihovny jsou instalovány do adresáře *MQ_INSTALLATION_PATH*\java\lib a 64bitové nativní knihovny lze najít v adresáři *MQ_INSTALLATION_PATH*\java\lib64 .

  Další informace o nativních knihovnách produktu IBM WebSphere MQ viz “Konfigurace knihoven JNI (Java Native Interface)” na stránce 701.
- Další skripty popsané v části “Skripty poskytnuté s třídami produktu WebSphere MQ pro službu JMS” na stránce 767. Tyto skripty jsou umístěny v adresáři *MQ_INSTALLATION_PATH*\java\bin.
- Specifikace tříd IBM WebSphere MQ pro rozhraní JMS API. Nástroj Javadoc byl použit ke generování stránek HTML obsahujících specifikace rozhraní API.

  Stránky HTML se nacházejí v adresáři *MQ_INSTALLATION_PATH*\java\doc\WMQJMSClasses.

  Na systémech UNIX, Linuxa Windows tento podadresář obsahuje jednotlivé stránky HTML.
- Podpora pro OGSi. Balíky OSGi jsou instalovány v adresáři java \lib\OSGi a jsou popsány v “Podpora pro OSGi” na stránce 768.

- Adaptér prostředků IBM WebSphere MQ , který lze implementovat do libovolného aplikačního serveru kompatibilního s JCA 1.5 (nebo novější).

  Adaptér prostředku IBM WebSphere MQ se nachází v adresáři *MQ_INSTALLATION_PATH*\java\lib\jca; další informace viz “Adaptér prostředků produktu IBM WebSphere MQ” na stránce 710 .
- V systému Windows jsou 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 WebSphere MQ . Jedná se o následující adresáře:

- Přenos protokolu SOAP produktu IBM WebSphere MQ , který poskytuje přenos JMS pro SOAP, je nainstalován do adresáře *MQ_INSTALLATION_PATH*\java\lib\soap. Další informace o přenosu protokolu SOAP v produktu IBM WebSphere MQ naleznete v sekci Informačního centra s popisem “Přenos WebSphere MQ pro SOAP” na stránce 914.
- Na distribuovaných platformách je produkt IBM WebSphere MQ Bridge for HTTP instalován v adresáři *MQ_INSTALLATION_PATH*\java\lib\http. Další informace o mostu IBM WebSphere MQ pro protokol HTTP naleznete v části Informačního centra s popisem produktu “Most WebSphere MQ pro protokol HTTP” na stránce 990 .

Některé ukázkové aplikace jsou dodávány s třídami produktu IBM WebSphere MQ pro platformu JMS. Tabulka 91 na stránce 698 zobrazuje, kde jsou ukázkové aplikace nainstalovány na každé platformě.

Tabulka 91. Adresáře ukázek

Platforma	**Adresář**
AIX	*MQ_INSTALLATION_PATH*/samp/jms
HP Integrity NonStop Server	*MQ_INSTALLATION_PATH*/opt/mqm/samp/jms
HP-UX, Linuxa Solaris	*MQ_INSTALLATION_PATH*/samp/jms
Windows	*MQ_INSTALLATION_PATH*\tools\jms.
MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM WebSphere MQ .	

Po instalaci možná budete muset provést některé konfigurační úlohy pro kompilaci a spuštění aplikací.

“Proměnné prostředí používané třídami produktu IBM WebSphere MQ pro platformu JMS” na stránce 699 popisuje cestu ke třídě vyžadovanou ke spuštění jednoduchých tříd IBM WebSphere MQ pro aplikace JMS. Toto téma také popisuje další soubory JAR, které je třeba odkazovat ve zvláštních situacích, a proměnné prostředí, které musíte nastavit ke spuštění skriptů poskytnutých s třídami produktu IBM WebSphere MQ pro platformu JMS.

Pokud potřebujete, aby aplikace třídy IBM WebSphere MQ pro aplikaci JMS odkazovali na kód zapsaný v jiných jazycích než Java (například při připojování k správci front), vysvětluje produkt “Konfigurace knihoven JNI (Java Native Interface)” na stránce 701 , kde najde umístění knihoven JNI (Java Native Interface), které lze zadat jako parametr příkazu jazyka Java.

Chcete-li ovládat vlastnosti, jako je například trasování a protokolování aplikace, musíte poskytnout soubor vlastností konfigurace. Soubor vlastností konfigurace IBM WebSphere MQ pro soubor vlastností JMS je popsán v části “Konfigurační soubor IBM WebSphere MQ classes for JMS” na stránce 703.

Instalace a upgrade tříd produktu WebSphere MQ pro soubory JAR platformy JMS

Jediný podporovaný způsob získání tříd produktu IBM WebSphere MQ pro soubory JAR platformy JMS do systému je instalovat produkt IBM WebSphere MQ nebo produkt WebSphere MQ V7.5 Clients SupportPac- MQC75nebo pomocí nástroje pro správu softwaru, jako je například produkt Apache Maven, pro více informací viz “IBM WebSphere MQ classes for JMS a nástroje pro správu softwaru” na stránce 705.

Nepřesouvejte nebo kopírujte třídy IBM WebSphere MQ pro soubory JAR JMS nebo nativní knihovny, do jiných počítačů nebo do jiného umístění na počítači, kde byly nainstalovány třídy IBM WebSphere MQ pro platformu JMS, pokud nepoužíváte nástroj pro správu softwaru.

- Opravné sady nelze aplikovat na "instalaci", kde byly soubory JAR zkopírovány z jiného počítače, protože tím je mnohem těžší zajistit, aby všechny soubory JAR byly udržovány v jednotlivých krocích, a jsou na kompatibilních úrovních.
- Kopírování tříd produktu IBM WebSphere MQ pro soubory JAR platformy JMS mezi počítači může také vést k více kopiím souborů umístěných na stejném počítači, což může způsobit problémy při opravě kódu a ladění problémů.
- Příkaz dspmqver , který se používá k zobrazení informací o verzi z instalace produktu IBM WebSphere MQ , zobrazuje pouze informace o verzi pro třídy IBM WebSphere MQ pro platformu JMS nainstalované do adresáře `\java\lib` .

  Pokud se více kopií souborů nachází na stejném počítači, spuštění produktu **dspmqver** nemusí poskytovat přesné informace o verzi tříd produktu IBM WebSphere MQ pro rozhraní JMS, které používá aplikace.

Nezahrnujte do archivů aplikací třídy IBM WebSphere MQ pro soubory JAR rozhraní JMS (například archivy podnikových aplikací nebo soubory EAR).

- Aktualizace tříd produktu IBM WebSphere MQ pro platformu JMS nelze použít s použitím opravné sady IBM WebSphere MQ Fix Pack.
- Podpora IBM nepodporuje možnost snadného určení verze tříd produktu IBM WebSphere MQ pro platformu JMS, které jsou používány aplikací.
- Problémy mohou nastat, pokud více aplikací spuštěných ve stejném běhovém prostředí Java zahrnuje různé verze tříd produktu IBM WebSphere MQ pro rozhraní JMS, protože více verzí tříd produktu IBM WebSphere MQ pro platformu JMS je načíteno do prostředí Java Runtime Environment současně.

  Příklady těchto problémů zahrnují následující výjimky:

```
java.lang.ClassCastException :
com.ibm.mq.jmqi.system.JmqiSystemEnvironment incompatible with
com.ibm.mq.jmqi.system.JmqiSystemEnvironment

java.lang.ClassCastException :
com.ibm.mq.jms.MQQueue incompatible with com.ibm.mq.jms.MQQueue
```

- Pokud aplikace používá přenos BINDINGS pro připojení ke správci front, všechny hlavní upgrady správce front také vyžadují aktualizaci aplikace, aby zahrnovala odpovídající úroveň tříd produktu IBM WebSphere MQ pro platformu JMS.

  Je-li například správce front upgradován na úroveň produktu IBM WebSphere MQ verze 7.5 , je třeba aktualizovat všechny aplikace, které se připojují ke správci front pomocí přenosu BINDINGS, aby zahrnovaly třídy produktu IBM WebSphere MQ verze 7.5 pro platformu JMS.

Proměnné prostředí používané třídami produktu IBM WebSphere MQ pro platformu JMS

Než budete moci kompilovat a spustit třídy IBM WebSphere MQ pro aplikace JMS, musí nastavení proměnné prostředí CLASSPATH obsahovat třídy IBM WebSphere MQ pro soubor JAR JMS (Java archive). V závislosti na vašich požadavcích budete možná muset do své cesty ke třídě přidat další soubory JAR. Chcete-li spustit skripty poskytnuté s třídami produktu IBM WebSphere MQ pro platformu JMS, musí být nastaveny další proměnné prostředí.

Chcete-li zkompilovat a spustit třídy IBM WebSphere MQ pro aplikace JMS, použijte nastavení CLASSPATH pro vaši platformu, jak je zobrazeno v Tabulka 92 na stránce 700. Nastavení zahrnuje adresář ukázek, takže můžete kompilovat a spustit třídy produktu IBM WebSphere MQ pro ukázkové aplikace JMS. Případně můžete zadat cestu ke třídě v příkazu **java** místo použití proměnné prostředí.

Tabulka 92. Nastavení CLASSPATH pro kompilaci a spuštění tříd produktu IBM WebSphere MQ pro aplikace JMS, včetně ukázkových aplikací	
Platforma	**Nastavení CLASSPATH**
AIX	CLASSPATH=*MQ_INSTALLATION_PATH*/java/lib/com.ibm.mqjms.jar: *MQ_INSTALLATION_PATH*/samp/jms:
HP Integrity NonStop Server	CLASSPATH=*MQ_INSTALLATION_PATH*/java/lib/com.ibm.mqjms.jar: *MQ_INSTALLATION_PATH*/samp/jms:
HP-UX, Linuxa Solaris	CLASSPATH=*MQ_INSTALLATION_PATH*/java/lib/com.ibm.mqjms.jar: *MQ_INSTALLATION_PATH*/samp/jms:
IBM i	CLASSPATH=/QIBM/ProdData/mqm/java/lib/com.ibm.mqjms.jar: /QIBM/ProdData/mqm/java/j/samples/jms:
Windows	CLASSPATH=*MQ_INSTALLATION_PATH*\java\lib\com.ibm.mqjms.jar; *MQ_INSTALLATION_PATH*\tools\jms;
z/OS	CLASSPATH=*MQ_INSTALLATION_PATH*/mqm/V7R0M0/java/lib/ com.ibm.mqjms.jar: *MQ_INSTALLATION_PATH*/mqm/V6R0M0/java/samples/jms:
MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM WebSphere MQ .	

Soubor typu manifest souboru JAR com.ibm.mqjms.jar obsahuje odkazy na většinu ostatních souborů JAR vyžadovaných třídami produktu IBM WebSphere MQ pro aplikace JMS, a proto nemusíte přidávat tyto soubory JAR do cesty ke třídám. Tyto soubory JAR zahrnují ty, které jsou vyžadovány aplikacemi, které používají rozhraní JNDI (Java Naming and Directory Interface) k načtení administrovaných objektů z adresářové služby a aplikací, které používají rozhraní JTA (Java Transaction API).

Do cesty ke třídě však musíte zahrnout další soubory JAR za následujících okolností:

- Pokud používáte třídy ukončení kanálu, které implementují rozhraní uživatelské procedury kanálu definovaná v balíku com.ibm.mq , místo těch, které jsou definovány v balíku com.ibm.mq.exits , musíte do své cesty ke třídě přidat třídy IBM WebSphere MQ pro soubor JAR Java com.ibm.mq.jar.
- Pokud kompilujete kód Java pomocí sady SDK (Java 2 Software Development Kit) ve verzi 1.4.2, je třeba do cesty ke třídě přidat následující soubory JAR:
  - jms.jar
  - com.ibm.mq.jmqi.jar

  Navíc, pokud vaše aplikace používá JNDI k načtení administrovaných objektů z adresářové služby, musíte do své cesty ke třídě přidat také následující soubory JAR:
  - fscontext.jar
  - jndi.jar
  - ldap.jar
  - providerutil.jar

  A pokud vaše aplikace používá JTA, musíte také přidat jta.jar do své cesty ke třídě.

  Všimněte si, že tyto další soubory JAR jsou nezbytné pouze pro kompilaci vašich aplikací, nikoli pro jejich spuštění.

Při kompilaci pomocí volby -Xlint se může zobrazit zpráva s varováním, že soubor com.ibm.mq.ese.jar není přítomen. Varování můžete ignorovat. Tento soubor je k dispozici pouze v případě, že jste nainstalovali produkt Extended Security Edition.

Skripty poskytnuté s třídami produktu IBM WebSphere MQ pro platformu JMS používají následující proměnné prostředí:

MQ_JAVA_DATA_PATH
Tato proměnná prostředí určuje adresář pro výstup protokolu a trasování.

INSTALAČNÍ_CESTA MQ_JAVA_INSTALL_PATH
Tato proměnná prostředí určuje adresář, v němž jsou instalovány třídy WebSphere MQ pro platformu JMS.

KOŘEN ROZHRANÍ MQ_JAVA_LIB_PATH
Tato proměnná prostředí určuje adresář, ve kterém jsou uloženy třídy WebSphere MQ pro knihovny JMS, jak je zobrazeno v Tabulka 93 na stránce 702.

V systému Windowsjsou všechny proměnné prostředí nastaveny automaticky během instalace. Na jakékoliv jiné platformě je musíte nastavit sami.

Chcete-li nastavit proměnné prostředí, používáte-li 32bitové prostředí JVM na systémech UNIX, HP Integrity NonStop Server, nebo Linux , můžete použít skript setjmsenv. Chcete-li nastavit proměnné prostředí při použití 64bitového prostředí JVM v systému UNIX nebo Linux , můžete použít skript setjmsenv64. Tyto skripty jsou uloženy v adresáři *MQ_INSTALLATION_PATH*`/java/bin` , kde *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM WebSphere MQ .

Skript setjmsenv nebo setjmsenv64 můžete použít různými způsoby: Můžete jej použít jako základ pro nastavení požadovaných proměnných prostředí, jak je zobrazeno v tabulce, nebo je můžete přidat do produktu `.profile` pomocí textového editoru. Máte-li netypické nastavení, upravte obsah skriptu podle potřeby. Případně můžete skript spustit v každé relaci, ze které mají být spouštěny spouštěcí skripty JMS. Vyberete-li tuto volbu, musíte spustit skript v každém okně shellu, který spustíte, během procesu verifikace služby JMS zadáním příkazu `. ./setjmsenv` nebo `. ./setjmsenv64`

Konfigurace knihoven JNI (Java Native Interface)

Třídy IBM WebSphere MQ pro aplikace JMS, které se buď připojují ke správci front s použitím přenosu vazeb, nebo které se připojují ke správci front pomocí přenosu klienta a používají výstupní programy kanálu napsané v jiných jazycích než Java, je třeba spustit v prostředí, které přistupuje ke knihovnám JNI (Java Native Interface).

Informace o této úloze

Chcete-li nastavit toto prostředí, musíte nakonfigurovat cestu ke knihovně prostředí tak, aby prostředí JVM (Java Virtual Machine) mohlo načíst knihovnu mqjbnd, než spustíte třídy produktu IBM WebSphere MQ pro aplikaci JMS.

Produkt IBM WebSphere MQ poskytuje dvě knihovny JNI (Java Native Interface):

mqjbnd
Tato knihovna je používána aplikacemi, které se připojují ke správci front pomocí přenosu vazeb. Poskytuje rozhraní mezi třídami IBM WebSphere MQ pro JMS a správce front. Knihovna mqjbnd nainstalovaná s produktem IBM WebSphere MQ verze 7.5 může být použita pro připojení k libovolnému správci front produktu IBM WebSphere MQ verze 7.5 (nebo starším).

mqjexitstub02
Knihovna mqjexitstub02 je načtena třídami produktu IBM WebSphere MQ pro JMS, když se aplikace připojuje ke správci front pomocí přenosu klienta a používá ukončovací program kanálu, který je napsán v jiném jazyce než Java.

Na určitých platformách instaluje produkt IBM WebSphere MQ 32bitové a 64bitové verze těchto knihoven JNI. Umístění knihoven pro každou platformu je zobrazeno v Tabulka 1

Tabulka 93. Umístění tříd produktu IBM WebSphere MQ pro knihovny platformy JMS pro každou platformu	
Platforma	**Adresář obsahující třídy IBM WebSphere MQ pro knihovny JMS**
AIX	*MQ_INSTALLATION_PATH*/java/lib (32bitové knihovny) *MQ_INSTALLATION_PATH*/java/lib64 (64 bitové knihovny)
HP-UX	*MQ_INSTALLATION_PATH*/java/lib (32bitové knihovny) *MQ_INSTALLATION_PATH*/java/lib64 (64 bitové knihovny)
Linux (Mocnina , x86-64 a zSeries s390x platformy)	*MQ_INSTALLATION_PATH*/java/lib (32bitové knihovny) *MQ_INSTALLATION_PATH*/java/lib64 (64 bitové knihovny)
Linux (platformax86) Linux (Platforma zSeries)	*MQ_INSTALLATION_PATH* /java/bin
Solaris (platformyx86-64 a SPARC)	*MQ_INSTALLATION_PATH*/java/lib (32bitové knihovny) *MQ_INSTALLATION_PATH*/java/lib64 (64 bitové knihovny)
Windows	*MQ_INSTALLATION_PATH*\java\lib (32bitové knihovny) *MQ_INSTALLATION_PATH*\java\lib64 (64 bitové knihovny)
MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM WebSphere MQ .	

Postup

1. Zkonfigurujte vlastnost **`java.library.path`** prostředí JVM, kterou lze provést dvěma způsoby:
   - Uvedením argumentu JVM, jak je zobrazeno v následujícím příkladu:

     ```
     -Djava.library.path=<path_to_library_directory>
     ```

     Linux Například pro 64bitové prostředí JVM v systému Linux pro instalaci výchozího umístění zadejte:

     ```
     -Djava.library.path=/opt/mqm/java/lib64
     ```

   - Konfigurací prostředí shellu tak, aby prostředí JVM nastavila své vlastní `java.library.path`. Tato cesta se liší podle platformy a umístění, do kterého jste nainstalovali produkt IBM WebSphere MQ. Například pro 64bitové prostředí JVM a pro výchozí umístění instalace produktu IBM WebSphere MQ můžete použít následující nastavení:

     AIX
     ```
     export LIBPATH=/usr/mqm/java/lib64:$LIBPATH
     ```

Solaris HP-UX Linux
```
export LD_LIBRARY_PATH=/opt/mqm/java/lib64:$LD_LIBRARY_PATH
```

Windows
```
set PATH=C:\Program Files\IBM\MQ\java\lib64;%PATH%
```

Příklad zásobníku výjimek, který vidíte v situaci, kdy prostředí nebylo správně nakonfigurováno, je následující:

```
Příčina: com.ibm.mq.jmqi.local.LocalMQ$4: CC=2;RC=2495;
AMQ8598: Nezdařilo se načíst nativní knihovnu JNI produktu WebSphere MQ : 'mqjbnd'.
    na adrese com.ibm.mq.jmqi.local.LocalMQ.loadLib(LocalMQ.java:1268)
    na adrese com.ibm.mq.jmqi.local.LocalMQ$1.run(LocalMQ.java:309)
    v java.security.AccessController.doPrivileged(AccessController.java:400)
    na adrese com.ibm.mq.jmqi.local.LocalMQ.initialise_inner(LocalMQ.java:259)
    na adrese com.ibm.mq.jmqi.local.LocalMQ.initialise(LocalMQ.java:221)
    na adrese com.ibm.mq.jmqi.local.LocalMQ. < init> (LocalMQ.java:1350)
    na adrese com.ibm.mq.jmqi.local.LocalServer. < init> (LocalServer.java:230)
    at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native metoda)
    at
sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:86)

    at
sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.jav
a:58).
    at java.lang.reflect.Constructor.newInstance(Constructor.java:542)
    na adrese com.ibm.mq.jmqi.JmqiEnvironment.getInstance(JmqiEnvironment.java:706)
    na adrese com.ibm.mq.jmqi.JmqiEnvironment.getMQI(JmqiEnvironment.java:640)
    na adrese
com.ibm.msg.client.wmq.factories.WMQConnectionFactory.createV7ProviderConnection(WMQConnectionF
actory.java:8437)
    ... 7 dalších
Příčina: java.lang.UnsatisfiedLinkError: mqjbnd (není nalezeno v java.library.path)
    at java.lang.ClassLoader.loadLibraryWithPath(ClassLoader.java:1235)
    na java.lang.ClassLoader.loadLibraryWithClassLoader(ClassLoader.java:1205)
    na java.lang.System.loadLibrary(System.java:534)
    na adrese com.ibm.mq.jmqi.local.LocalMQ.loadLib(LocalMQ.java:1240)
    ... 20 dalších
```

2. Po nastavení 32bitového nebo 64bitového prostředí spusťte třídy produktu IBM WebSphere MQ pro aplikaci JMS pomocí následujícího příkazu:

```
java application-name
```

, kde *název-aplikace* je název tříd produktu IBM WebSphere MQ pro aplikaci JMS, která má být spuštěna.

Výjimku obsahující kód příčiny IBM WebSphere MQ 2495 (MQRC_MODULE_NOT_FOUND) vyvolá třídy produktu IBM WebSphere MQ pro platformu JMS, pokud:

- Třídy IBM WebSphere MQ pro aplikaci JMS se spouštějí v 32bitovém prostředí JRE a 64bitové prostředí bylo nastaveno pro třídy IBM WebSphere MQ pro JMS, protože 32bitové prostředí JRE nemůže načíst 64bitovou knihovnu Java Native Library.
- Třídy IBM WebSphere MQ pro aplikaci JMS jsou spouštěny v 64bitovém běhovém prostředí Java a 32bitové prostředí bylo nastaveno pro třídy IBM WebSphere MQ pro službu JMS, protože 64bitové prostředí JRE (Java Runtime Environment) nemůže načíst 32bitovou nativní knihovnu Java.

Konfigurační soubor IBM WebSphere MQ classes for JMS

Konfigurační soubor tříd produktu WebSphere MQ pro službu JMS určuje vlastnosti, které se používají ke konfiguraci tříd produktu WebSphere MQ pro platformu JMS.

Formát konfiguračního souboru WebSphere MQ pro konfigurační soubor JMS je formát standardního souboru vlastností Java. Ukázkový konfigurační soubor s názvem jms.config je dodáván v podadresáři `bin` instalačního adresáře WebSphere MQ classes for JMS. Tento soubor dokumentuje všechny podporované vlastnosti a jejich výchozí hodnoty.

Můžete zvolit název a umístění konfiguračního souboru produktu WebSphere MQ pro soubor konfigurace JMS. Když spouštíte aplikaci, použijte příkaz jazyka **java** s následujícím formátem:

```
java -Dcom.ibm.msg.client.config.location=config_file_url application_name
```

V příkazu *config_file_url* je adresa URL (Uniform Resource Locator), která určuje název a umístění konfiguračního souboru WebSphere MQ pro konfigurační soubor JMS. Podporovány jsou adresy URL následujících typů: http, file, ftp, and jar.

Zde je příklad příkazu **java** :

```
java -Dcom.ibm.msg.client.config.location=file:/D:/mydir/myjms.config MyAppClass
```

Tento příkaz identifikuje konfigurační soubor WebSphere MQ pro konfigurační soubor JMS jako soubor D:\mydir\mjms.config na lokálním systému Windows .

Při spuštění aplikace načte třídy produktu WebSphere MQ pro službu JMS obsah konfiguračního souboru a uloží zadané vlastnosti do interního úložiště vlastností. Pokud příkaz **java** neidentifikuje konfigurační soubor, nebo pokud nelze konfigurační soubor nalézt, třídy WebSphere MQ pro JMS použijí výchozí hodnoty pro všechny vlastnosti. V případě potřeby můžete přepsat jakoukoli vlastnost v konfiguračním souboru tak, že ji uvedete jako systémovou vlastnost v příkazu **java** .

Konfigurační soubor tříd produktu WebSphere MQ pro konfigurační soubor JMS lze použít s libovolným podporovaným transportem mezi aplikací a správcem front nebo zprostředkovatelem.

Všimněte si, že nemůžete zadat trasování spuštění nastavením vlastnosti v konfiguračním souboru WebSphere MQ pro soubor konfigurace JMS. Trasování spouštění lze určit pouze nastavením systémové vlastnosti v příkazu **java** , jak je uvedeno v následujícím příkladu:

```
java -Dcom.ibm.msg.client.commonservices.trace.startup=true
     -Dcom.ibm.msg.client.config.location=file:/D:/mydir/myjms.config
      MyAppClass
```

Potlačení vlastností uvedených v konfiguračním souboru klienta WebSphere MQ MQI

Konfigurační soubor klienta WebSphere MQ MQI může také určovat vlastnosti, které se používají ke konfiguraci tříd produktu WebSphere MQ pro službu JMS. Vlastnosti zadané v konfiguračním souboru klienta WebSphere MQ MQI se však používají pouze v případě, že se aplikace připojuje ke správci front v režimu klienta.

V případě potřeby můžete přepsat libovolný atribut v konfiguračním souboru klienta WebSphere MQ MQI tak, že jej uvedete jako vlastnost v konfiguračním souboru WebSphere MQ pro konfigurační soubor JMS. Chcete-li přepsat atribut v konfiguračním souboru klienta WebSphere MQ MQI, použijte položku s následujícím formátem v souboru WebSphere MQ pro konfigurační soubor JMS:

```
com.ibm.mq.cfg.stanza.propName=propValue
```

Proměnné v položce mají následující význam:

sekce
Název oddílu v konfiguračním souboru klienta WebSphere MQ MQI, který obsahuje atribut

propName
Název atributu, jak je uveden v konfiguračním souboru klienta WebSphere MQ MQI

propValue
Hodnota vlastnosti potlačující hodnotu atributu určeného v konfiguračním souboru klienta WebSphere MQ MQI

Případně můžete přepsat atribut v konfiguračním souboru klienta WebSphere MQ MQI tak, že uvedete vlastnost jako systémovou vlastnost v příkazu **java** . Chcete-li určit vlastnost jako systémovou vlastnost, použijte předchozí formát.

Pouze následující atributy v konfiguračním souboru klienta WebSphere MQ MQI jsou relevantní pro třídy produktu WebSphere MQ pro platformu JMS. Pokud uvedete nebo přepisují jiné atributy, nemá žádný efekt.

sekce	Atribut
ClientExitSekce Cesta ke konfiguračnímu souboru klienta	ExitsDefaultPath
ClientExitSekce Cesta ke konfiguračnímu souboru klienta	ExitsDefaultPath64
ClientExitSekce Cesta ke konfiguračnímu souboru klienta	Cesta ke třídě JavaExits
stanzaMessageBuffer konfiguračního souboru klienta	MaximumSize
stanzaMessageBuffer konfiguračního souboru klienta	PurgeTime
stanzaMessageBuffer konfiguračního souboru klienta	UpdatePercentage
Sekce TCP konfiguračního souboru klienta	ClntRcvBufSize
Sekce TCP konfiguračního souboru klienta	ClntSndBufSize
Sekce TCP konfiguračního souboru klienta	Časový limit připojení
Sekce TCP konfiguračního souboru klienta	KeepAlive

IBM WebSphere MQ classes for JMS a nástroje pro správu softwaru

Nástroje pro správu softwaru, jako např. Apache Maven, lze použít s produktem IBM WebSphere MQ classes for JMS.

Řada velkých rozvojových organizací používá tyto nástroje k centrální správě úložišť knihoven jiných dodavatelů.

IBM WebSphere MQ classes for JMS se skládá z určitého počtu souborů JAR. Když vyvíjíte aplikace v jazyce Java pomocí tohoto rozhraní API, je na počítači, kde je aplikace vyvíjována, vyžadována instalace serveru IBM WebSphere MQ , klienta nebo klienta SupportPac .

Chcete-li použít takový nástroj a přidat soubory JAR, které tvoří IBM WebSphere MQ classes for JMS do centrálně spravovaného úložiště, musí být dodrženy následující body:

- Úložiště nebo kontejner musí být k dispozici pouze pro vývojáře v rámci vaší organizace. Jakékoli rozdělení mimo organizaci není povoleno.
- Úložiště musí obsahovat úplnou a konzistentní sadu souborů JAR z jedné verze produktu IBM WebSphere MQ nebo z opravné sady Fix Pack.
- Jste zodpovědní za aktualizaci úložiště pomocí jakékoli údržby poskytované společností IBM Support.

Pro produkt IBM WebSphere MQ Version 7.5musí být do úložiště nainstalovány následující soubory JAR:

- com.ibm.mqjms.jar.
- com.ibm.mq.jar
- com.ibm.mq.jmqi.jar
- com.ibm.mq.pcf.jar
- com.ibm.mq.headers.jar
- Soubor CL3Export.jar je povinný, pokud používáte produkt IBM WebSphere MQ classes for JMS.
- Hodnota CL3Nonexport.jar je povinná, pokud používáte produkt IBM WebSphere MQ classes for JMS.
- Příkaz jndi.jar je povinný, pokud používáte produkt IBM WebSphere MQ classes for JMS.

- Soubor ldap.jar je povinný, pokud používáte produkt IBM WebSphere MQ classes for JMS.
- Příkaz rmm.jar je povinný, pokud používáte produkt IBM WebSphere MQ classes for JMS.
- Příkaz dhbcore.jar je povinný, pokud používáte produkt IBM WebSphere MQ classes for JMS.
- Hodnota jms.jar je povinná, pokud používáte produkt IBM WebSphere MQ classes for JMS.
- Soubor fscontext.jar je povinný, pokud používáte produkt IBM WebSphere MQ classes for JMS a přistupujete ke spravovaným objektům služby JMS, které jsou uloženy v kontextu rozhraní JNDI systému souborů.
- providerutil.jar používáte-li produkt IBM WebSphere MQ classes for JMS a přístup ke spravovaným objektům služby JMS, které jsou uloženy v kontextu rozhraní JNDI systému souborů.

Spuštění tříd produktu WebSphere MQ pro aplikace JMS v rámci správce zabezpečení produktu Java

Třídy WebSphere MQ pro platformu JMS lze spustit s povoleným správcem zabezpečení produktu Java . Chcete-li úspěšně spustit aplikace s povoleným správcem zabezpečení, musíte nakonfigurovat prostředí JVM (Java virtual machine) s vhodným konfiguračním souborem zásad.

Nejjednodušším způsobem, jak to provést, je změnit konfigurační soubor zásad dodaný spolu s vaším prostředím JRE. Na většině systémů je tento soubor uložen v cestě `lib/security/java.policy`, relativně k adresáři JRE. Konfigurační soubor zásad můžete upravit pomocí svého preferovaného editoru nebo pomocí programu `policytool` dodaného s vaším prostředím JRE.

Důležité: V 7.5.0.8 Termín *allowlist* nahradil termín *whitelist*. Jednou z výjimek je následující názvy systémových vlastností produktu Java .

Použijete-li ke své aplikaci mechanizmus správce zabezpečení produktu Java , musíte udělit následující oprávnění:

- FilePermission na libovolném souboru allowlist, který používáte, s oprávněním ke čtení pro režim ENFORCEMENT, zapište oprávnění pro režim DISCOVER.
- PropertyPermission (čtení) na vlastnostech com.ibm.mq.jms.whitelist, com.ibm.mq.jms.whitelist.discovera com.ibm.mq.jms.whitelist.mode .

Parametr allowlisting ClassName je podporován s APAR IT14385 a IBM WebSphere MQ Version 7.5.0, opravná sada Fix Pack 8. Další informace viz “ClassName allowlisting v JMS ObjectMessage” na stránce 707.

Zde je příklad dvou položek v konfiguračním souboru zásad, které umožňují úspěšné spuštění tříd produktu WebSphere MQ pro platformu JMS pod výchozím správcem zabezpečení:

```
grant codeBase "file:MQ_INSTALLATION_PATH/java/lib/com.ibm.mq.jmqi.jar" {
  //Required
  permission java.util.PropertyPermission "user.name","read";
  permission java.util.PropertyPermission "os.name","read";
  //Required if mqclient.ini/mqs.ini configuration files are used
  permission java.io.FilePermission "/var/mqm/mqclient.ini","read";
  permission java.io.FilePermission "/var/mqm/mqs.ini","read";
  //For the client transport type.
  permission java.net.SocketPermission "*","connect";
  //For the bindings transport type.
  permission java.lang.RuntimePermission "loadLibrary.*";
  //For applications that use CCDT tables (access to the CCDT
AMQCLCHL.TAB)
  permission java.io.FilePermission
"/var/mqm/qmgrs/QMGR/@ipcc/AMQCLCHL.TAB","read";
  //For applications that use User Exits
  permission java.io.FilePermission "/var/mqm/exits/*","read";
  permission java.lang.RuntimePermission "createClassLoader";
  //Required for the z/OS platform
  permission java.util.PropertyPermission
"com.ibm.vm.bitmode","read";
};
grant codeBase "file:MQ_INSTALLATION_PATH/java/lib/com.ibm.mqjms.jar" {
  permission java.util.PropertyPermission "user.name","read";
  permission java.util.PropertyPermission "os.name","read";
  permission java.util.PropertyPermission "console.encoding","read";
```

```
  permission java.lang.RuntimePermission "setContextClassLoader";
  //tracing permissions
  permission java.util.PropertyPermission "com.ibm.msg.client.commonservices.*","read";
  permission java.util.PropertyPermission "MQJMS_TRACE_LEVEL","read";
  permission java.util.logging.LoggingPermission "control";
  //Wherever trace output is expected
  permission java.io.FilePermission "/tmp/*","read,write";
  //Required for the z/OS platform
  permission java.util.PropertyPermission
"com.ibm.vm.bitmode","read";
};
```

Notes:

- Produkt *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.
- První příkaz `grant` obsahuje oprávnění vyžadovaná třídami WebSphere MQ pro službu JMS a druhý příkaz `grant` obsahuje oprávnění vyžadovaná třídami produktu WebSphere MQ pro aplikaci JMS.
- Chcete-li povolit třídám produktu WebSphere MQ pro službu JMS přístup k archivním souborům produktu Java (JAR) aplikace, přidejte následující oprávnění k prvnímu příkazu `grant` :

```
permission java.io.FilePermission "/path_to_your_app/-", "read";
```

- Chcete-li použít tyto příkazy `grant` v konfiguračním souboru zásad, může být třeba upravit názvy cest v závislosti na tom, kam jste nainstalovali třídy WebSphere MQ pro systém JMS a kam ukládáte vaše aplikace.
- Správci zabezpečení nepovolují ukázkové aplikace dodávané s třídami produktu WebSphere MQ pro systém JMS a skripty pro jejich spuštění.

V 7.5.0.8 ClassName allowlisting v JMS ObjectMessage

V 7.5.0.8 Ve třídách WebSphere MQ pro platformu JMS poskytuje podpora povolení seznamu tříd v rámci implementace rozhraní JMS ObjectMessage potenciální zmírnění rizik zabezpečení, která se potenciálně vztahují k serializaci objektů Java a k deserializaci.

Poznámka: Termín *allowlist* nahradil termín *whitelist*. Jedinou výjimkou jsou názvy systémové vlastnosti Java zmíněné v tomto tématu.

Mechanizmus serializace a deserializace objektů produktu Java byl identifikován jako potenciální riziko zabezpečení, protože deserializace vytváří instanci libovolných objektů produktu Java , kde je potenciál pro nekalciózně odeslaná data způsobovat různé problémy. Jedna pozoruhodná aplikace serializace je v ObjectMessages produktu Java Message Service (JMS), která používá serializaci pro zapouzdření a přenos libovolných objektů.

Serializační povolení je potenciální zmírnění některých rizik, která serializace představuje. Explicitně určením, které třídy lze zapouzdřovat a extrahovat z ObjectMessages, umožňuje povolení některé ochrany proti některým serializačním rizikům.

Allowlisting v třídách WebSphere MQ pro JMS

Při použití Oprava APAR IT14385 a IBM WebSphere MQ Version 7.5.0, opravná sada Fix Pack 8, třídy WebSphere MQ pro JMS podporují povolení seznamu tříd v implementaci rozhraní JMS ObjectMessage . Seznam allowlist definuje, které třídy produktu Java mohou být serializovány pomocí ObjectMessage.setObject() a deserializovány s objekty ObjectMessage.getObject().

Pokusy o serializaci nebo deserializaci instance třídy, která není zahrnuta v seznamu allowlist s parametrem ObjectMessage , způsobí vygenerování výjimky javax.jms.MessageFormatException a její příčinou je výjimka java.io.InvalidClassException .

Vytvoření seznamu povolených položek

Důležité: Třídy WebSphere MQ pro platformu JMS nelze distribuovat s použitím seznamu allowlist. Volba tříd, které mají být přeneseny pomocí volby ObjectMessages , je volba návrhu aplikace a produkt IBM WebSphere MQ jej nemůže předjímat.

Z tohoto důvodu umožňuje mechanizmus allowlisting pro dva režimy provozu:

Zjišťování
V tomto režimu vytvoří mechanizmus výpis úplných názvů tříd, které hlásí všechny třídy, které byly sledovány, aby byly serializovány nebo deserializovány v ObjectMessages.

Vynucení
V tomto režimu mechanizmus vynucuje povolení, odmítá pokusy o serializaci nebo deserializaci tříd, které nejsou v seznamu povolených položek.

Chcete-li tento mechanismus použít, musíte nejprve spustit příkaz DISCOVERY a shromáždit seznam aktuálně serializovaných a deserializovaných tříd, přezkoumat seznam a použít jej jako základ pro seznam povolených operací. Možná bude vhodné použít seznam nezměněný, ale před tím, než se rozhodnete toto provést, musí být seznam přezkoumán.

Řízení mechanismu allowlisting

K dispozici jsou tři systémové vlastnosti pro ovládání mechanizmu allowlisting:

com.ibm.mq.jms.whitelist
Tato vlastnost může být zadána jedním z následujících způsobů:

- Název cesty k souboru, který obsahuje seznam povolených, ve formátu identifikátoru URI souboru (který je počínaje `file:`). V režimu DISCOVERY je tento soubor zapisován mechanizmem allowlisting. Soubor nesmí existovat. Pokud soubor existuje, mechanizmus vygeneruje výjimku a nepřepisuje jej. V režimu ENFORCEMENT je tento soubor přečten mechanismem allowlisting.
- Čárkami oddělené plně kvalifikované názvy tříd, které tvoří seznam povolených.

Není-li tato vlastnost nastavena na nenastavená vlastnost, mechanizmus allowlist je neaktivní.

Pokud používáte správce zabezpečení produktu Java , je třeba zajistit, aby třídy WebSphere MQ pro soubory JAR platformy JMS měly přístup pro čtení a zápis do tohoto souboru.

com.ibm.mq.jms.whitelist.discover

- Je-li tato vlastnost nenastavena nebo nastavena na hodnotu false, bude mechanizmus allowlist spuštěn v režimu ENFORCEMENT.
- Je-li tato vlastnost nastavena na hodnotu true a jako identifikátor URI souboru byl určen parametr allowlist, bude mechanizmus allowlist spuštěn v režimu DISCOVERY.
- Je-li tato vlastnost nastavena na hodnotu true a byl-li jako seznam názvů tříd zadán seznam povolených názvů, aktivuje mechanizmus allowlist vhodnou výjimku.
- Je-li tato vlastnost nastavena na hodnotu true a seznam allowlist nebyl určen pomocí vlastnosti com.ibm.mq.jms.whitelist, mechanizmus allowlist je neaktivní.
- Je-li tato vlastnost nastavena na hodnotu true a soubor allowlist již existuje, mechanizmus allowlist vyvolá výjimku java.io.InvalidClassException a položky nebudou přidány do souboru.

com.ibm.mq.jms.whitelist.mode
Tato řetězcová vlastnost může být zadána libovolným ze tří způsobů:

- Je-li tato vlastnost nastavena na SERIALIZE, pak režim ENFORCEMENT provede ověření allowlist pouze v metodě ObjectMessage.setObject().
- Je-li tato vlastnost nastavena na hodnotu DESERIALIZE, pak režim ENFORCEMENT provádí ověřování allowlist pouze v metodě ObjectMessage.getObject().
- Je-li tato vlastnost nenastavena nebo nastavena na jakoukoli jinou hodnotu, pak režim ENFORCEMENT provede ověření povolení na obou metodách ObjectMessage.getObject() a ObjectMessage.setObject().

Formát souboru allowlist

Toto jsou hlavní funkce ve formátu souboru allowlist:

- Soubor allowlist je ve výchozím kódování souboru platformy s čárovým koncovkami odpovídajícího platformům.

  Poznámka: Pokud se soubor přesunujete mezi heterogenními systémy, může dojít ke konverzi souboru.
- Každý není-prázdný řádek obsahuje úplný název třídy. Prázdné řádky se ignorují.
- Komentáře lze zahrnout-cokoli následujícího po znaku '#', na konec řádku, se ignoruje.
- Existuje velmi základní mechanizmus wildsorting:
  - '*' může být **poslední** prvek názvu třídy.
  - Hodnota '*' odpovídá prvku **jediný** názvu třídy, tj. třídě, ale ne součástí balíku.

  Takže `com.ibm.mq.*` by odpovídalo `com.ibm.mq.MQMessage` , ale ne `com.ibm.mq.jmqi.remote.api.RemoteFAP`.

  Použití zástupných znaků nepracuje pro třídy ve výchozím balíku, které jsou určeny pro třídy bez explicitního názvu balíku, takže název třídy "*" je odmítnut.
- Chybně naformátované soubory seznamu povolených souborů, například soubory obsahující položku jako `com.ibm.mq.*.Message`, kde zástupný znak není posledním prvkem, je výsledkem vyvolání výjimky java.lang.IllegalArgumentException .
- Prázdný soubor allowlist má vliv na úplné zablokování použití ObjectMessage.

Formát seznamu allowList jako seznam oddělený čárkami

Stejný mechanizmus wildcars je k dispozici pro seznam povolených položek jako seznam oddělený čárkami.

- Systém '*' může být rozšířen operačním systémem, pokud je zadán na příkazovém řádku nebo ve skriptu shellu nebo dávkovém souboru, takže může vyžadovat speciální zacházení.
- Znak komentáře '#' lze použít pouze v případě, že je zadán soubor. Je-li seznam povolených názvů zadán jako seznam názvů tříd oddělených čárkami, předpokládá se, že operační systém nebo shell jej nezpracují, protože se jedná o výchozí znak komentáře v mnoha shellech systému UNIX nebo Linux , je považován za normální znak.

Kdy se stane objekt allowlisting?

Allowlisting je zahájen, když aplikace poprvé spustí metodu ObjectMessage setMessage() nebo getMessage().

Systémové vlastnosti jsou vyhodnoceny, soubor allowlist je otevřen a v režimu ENFORCEMENT, seznam povolených tříd jsou načteny, když je mechanismus inicializován. V tomto okamžiku je položka zapsána do souboru protokolu produktu IBM WebSphere MQ JMS pro danou aplikaci.

Když je mechanismus inicializován, jeho parametry nemusí být změněny. Doba inicializace není snadno předpovězena, protože závisí na chování aplikace. Nastavení vlastností systému a obsah souboru allowlist by proto měly být považovány za pevné od okamžiku spuštění aplikace. Neměňte vlastnosti nebo obsah souboru allowlist, když je aplikace spuštěna, protože výsledky nejsou garantovány.

Body k posouzení

Nejlepším přístupem ke zmírnění rizik, která jsou součástí serializace Java , by bylo prozkoumání alternativních přístupů k přenosu dat, jako je například použití JSON namísto ObjectMessage. Pomocí mechanizmů produktu IBM WebSphere MQ Advanced Message Security (AMS) můžete přidat další zabezpečení tím, že zajistíte, že zprávy pocházejí z důvěryhodných zdrojů.

Použijete-li ke své aplikaci mechanizmus správce zabezpečení produktu Java , musíte udělit následující oprávnění:

- FilePermission na libovolném souboru allowlist, který používáte, s oprávněním ke čtení pro režim ENFORCEMENT, zapište oprávnění pro režim DISCOVER.
- PropertyPermission (čtení) na vlastnostech com.ibm.mq.jms.whitelist, com.ibm.mq.jms.whitelist.discovera com.ibm.mq.jms.whitelist.mode .

Související pojmy

"Spuštění tříd produktu WebSphere MQ pro aplikace JMS v rámci správce zabezpečení produktu Java" na stránce 706

Třídy WebSphere MQ pro platformu JMS lze spustit s povoleným správcem zabezpečení produktu Java . Chcete-li úspěšně spustit aplikace s povoleným správcem zabezpečení, musíte nakonfigurovat prostředí JVM (Java virtual machine) s vhodným konfiguračním souborem zásad.

Adaptér prostředků produktu IBM WebSphere MQ

Adaptér prostředků umožňuje aplikacím spuštěným na aplikačním serveru přístup k prostředkům produktu IBM WebSphere MQ . Podporuje příchozí a odchozí komunikaci.

The Java Platform, Enterprise Edition (Java EE) Connector Architecture (JCA) provides a standard way of connecting applications running in a Java EE environment to an Enterprise Information System (EIS) such as IBM WebSphere MQ or Db2. Adaptér prostředků IBM WebSphere MQ implementuje rozhraní JCA 1.5 a obsahuje IBM WebSphere MQ classes for JMS. Umožňuje aplikacím JMS a objektům typu message-driven bean (MDB) spuštěným na aplikačním serveru přistupovat k prostředkům správce front produktu IBM WebSphere MQ . Adaptér prostředků podporuje doménu typu point-to-point i doménu publikování/odběru.

Adaptér prostředků IBM WebSphere MQ podporuje dva typy komunikace mezi aplikací a správcem front:

Odchozí komunikace
Aplikace spustí připojení ke správci front a poté odesílá zprávy JMS do míst určení rozhraní JMS a přijímá zprávy JMS z míst určení rozhraní JMS synchronním způsobem.

Příchozí komunikace
Zpráva JMS přicházející do místa určení rozhraní JMS se doručí do objektu MDB, který asynchronně zpracuje zprávu.

Další informace o produktu IBM WebSphere MQ classes for JMS najdete v tématu "Použití tříd produktu WebSphere MQ pro službu JMS" na stránce 692.

Adaptér prostředku také obsahuje IBM WebSphere MQ classes for Java. Třídy jsou automaticky dostupné pro aplikace spuštěné na aplikačním serveru, na který byl implementován adaptér prostředků, a umožněte aplikacím spuštěným na aplikačním serveru používat rozhraní API produktu IBM WebSphere MQ classes for Java při přístupu k prostředkům správce front produktu IBM WebSphere MQ . Další informace o produktu IBM WebSphere MQ classes for Java naleznete v tématu "Použití tříd produktu WebSphere MQ pro prostředí Java" na stránce 631.

Použití produktu IBM WebSphere MQ classes for Java v prostředí Java EE je podporováno s omezeními. Informace o těchto omezeních viz "Spuštění tříd produktu IBM WebSphere MQ pro aplikace Java v rámci platformy Java Enterprise Edition" na stránce 690.

Další požadovaná dokumentace pro podporu adaptéru prostředků JCA

Informace o tom, jak konfigurovat adaptér prostředků JCA, naleznete v dokumentaci k aplikačnímu serveru.

Každý aplikační server poskytuje svou vlastní sadu rozhraní administrace. Některé aplikační servery poskytují grafická uživatelská rozhraní pro definování prostředků JCA, ale ostatní vyžadují, aby administrátor zapsal plány implementace XML. Proto je mimo rozsah této dokumentace informace o tom, jak konfigurovat adaptér prostředků WebSphere MQ pro každý aplikační server. Tato dokumentace se zaměřuje pouze na to, co je třeba nakonfigurovat. Informace o konfiguraci adaptéru prostředků JCA najdete v dokumentaci dodané s aplikačním serverem.

Chcete-li porozumět této dokumentaci, musíte být obeznámeni s třídami JMS a WebSphere MQ pro platformu JMS. Mnohé z vlastností používaných ke konfiguraci adaptéru prostředků WebSphere MQ jsou ekvivalentní vlastnostem tříd WebSphere MQ pro objekty JMS a mají stejnou funkci.

Instalace adaptéru prostředků produktu WebSphere MQ

Adaptér prostředků produktu WebSphere MQ je dodáván jako soubor archivu prostředků (RAR). Nainstalujte soubor RAR na aplikační server. Možná budete muset přidat adresáře do systémové cesty.

Adaptér prostředků produktu WebSphere MQ je dodáván jako archivní soubor prostředku (RAR) s názvem wmq.jmsra.rar. Tento soubor je instalován s třídami produktu WebSphere MQ pro platformu JMS v adresáři zobrazeném v tématu Tabulka 94 na stránce 711.

Tabulka 94. Adresář obsahující soubor wmq.jmsra.rar pro jednotlivé platformy

Platforma	**Adresář**
AIX, HP-UX, Linux a Solaris	*MQ_INSTALLATION_PATH* /java/lib/jca
IBM i	/QIBM/ProdData/mqm/java/lib/jca
Windows	*MQ_INSTALLATION_PATH* \java\lib\jca.
Produkt *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.	

Soubor RAR obsahuje třídy WebSphere MQ pro JMS a implementaci rozhraní JCA v produktu WebSphere MQ .

Soubor RAR adaptéru prostředků produktu WebSphere MQ je třeba nainstalovat na aplikační server, ale to, jak to uděláte, závisí na aplikačním serveru. Informace o tom, jak instalovat soubor RAR adaptéru prostředků, naleznete v dokumentaci k aplikačnímu serveru.

Pro vázání vazeb na systémech UNIX and Linux se musíte ujistit, že adresář obsahující knihovny JNI (Java Native Interface) je v systémové cestě. Informace o umístění tohoto adresáře, který obsahuje také třídy WebSphere MQ pro knihovny JMS, viz Tabulka 93 na stránce 702. V systému Windowsje tento adresář automaticky přidán do systémové cesty během instalace tříd produktu WebSphere MQ pro službu JMS.

Transakce jsou podporovány v režimu klienta i vazby.

Adaptér prostředků produktu WebSphere MQ a verze tříd produktu WebSphere MQ pro službu JMS používané adaptérem prostředků musí být na stejné úrovni verze.

WebSphere Application Server a adaptér prostředků WebSphere MQ
Nepoužívejte adaptér prostředků WebSphere MQ s produktem WebSphere Application Server verze 6. Produkt WebSphere Application Server, V7, obsahuje verzi adaptéru prostředků WebSphere MQ V7 .

Nepoužívejte adaptér prostředků WebSphere MQ v rámci serveru WebSphere Application Server, V6. Chcete-li získat přístup k prostředkům správce front WebSphere MQ z aplikace JMS z produktu WebSphere Application Server, použijte poskytovatele systému zpráv WebSphere MQ . Poskytovatel systému zpráv produktu WebSphere MQ obsahuje verzi tříd produktu WebSphere MQ pro službu JMS.

Produkt WebSphere Application Server, V7, obsahuje verzi adaptéru prostředků WebSphere MQ V7 .

Další informace naleznete v technické poznámce Jaká verze adaptéru prostředků WebSphere MQ (RA) je dodávána se serverem WebSphere Application Server ?.

WebSphere Application Server Liberty a adaptér prostředků IBM WebSphere MQ
Adaptér prostředků produktu IBM WebSphere MQ Version 7.5 lze instalovat do produktu WebSphere Application Server Liberty verze 8.5.5, opravná sada 2 nebo novější, pomocí funkce wmqJmsClient-1.1 . Případně můžete na základě určitých omezení nainstalovat adaptér prostředků pomocí generické podpory Enterprise Edition Connector Architecture (Java EE JCA) produktu Java .

Obecná omezení při instalaci adaptéru prostředků do profilu Liberty

Následující omezení platí pro adaptér prostředků produktu Version 7.5 při použití funkce wmqJmsClient-1.1 a také při použití generické podpory architektury JCA:

- Produkt IBM WebSphere MQ classes for Java není podporován v Liberty. Nesmí se používat buď s funkcí systému zpráv produktu IBM WebSphere MQ Liberty, nebo s generickou podporou JCA. Další informace naleznete v tématu Použití rozhraní Java produktu WebSphere MQ v prostředí J2EE/JEE Environments.
- Adaptér prostředku IBM WebSphere MQ má typ transportu BINDINGS_THEN_CLIENT. Tento typ přenosu není podporován v rámci funkce systému zpráv produktu IBM WebSphere MQ Liberty.
- Funkce IBM WebSphere MQ Advanced Message Security (IBM WebSphere MQ AMS) není zahrnuta do funkce systému zpráv produktu IBM WebSphere MQ Liberty.

Adaptér prostředku IBM WebSphere MQ Version 7.5 nelze použít spolu s funkcí wmqJmsClient-2.0 .

Konfigurace adaptéru prostředků WebSphere MQ

Chcete-li konfigurovat adaptér prostředků produktu WebSphere MQ , definujete různé prostředky JCA a systémové vlastnosti.

Definujte prostředky JCA v následujících kategoriích:

- Vlastnosti objektu ResourceAdapter , které reprezentují globální vlastnosti adaptéru prostředků, například úroveň trasování diagnostiky. Tyto vlastnosti jsou popsány v tématu “Konfigurace objektu ResourceAdapter” na stránce 713.
- Vlastnosti objektu ActivationSpec , které určují, jak je objekt MDB aktivován pro příchozí komunikaci. Tyto vlastnosti jsou popsány v tématu “Konfigurace adaptéru prostředků pro příchozí komunikaci” na stránce 714.
- Vlastnosti objektu ConnectionFactory , které aplikační server používá k vytvoření objektu ConnectionFactory rozhraní JMS pro odchozí komunikaci. Tyto vlastnosti jsou popsány v tématu “Konfigurace adaptéru prostředků pro odchozí komunikaci” na stránce 727.
- Vlastnosti spravovaného objektu místa určení, které aplikační server používá k vytvoření objektu fronty JMS nebo objektu tématu JMS pro odchozí komunikaci. Tyto vlastnosti jsou také popsány v tématu “Konfigurace adaptéru prostředků pro odchozí komunikaci” na stránce 727.

Soubor RAR adaptéru prostředků produktu WebSphere MQ obsahuje soubor s názvem META-INF/ra.xmlobsahující deskriptor zavedení pro adaptér prostředků. Tento deskriptor implementace je definován schématem XML v https://java.sun.com/xml/ns/j2ee/connector_1_5.xsd a obsahuje informace o adaptéru prostředků a službách, které poskytuje. Aplikační server může také vyžadovat plán implementace pro adaptér prostředků. Tento plán implementace je specifický pro aplikační server. Například produkt WebSphere Application Server Community Edition vyžaduje plán implementace s názvem geronimo-ra.xml.

Používáte-li zabezpečení SSL (Secure Sockets Layer), určete umístění souboru úložiště klíčů a souboru úložiště údajů o důvěryhodnosti jako systémové vlastnosti prostředí JVM, jak je uvedeno v následujícím příkladu:

```
java ... -Djavax.net.ssl.keyStore=
key_store_location
         -Djavax.net.ssl.trustStore=trust_store_location
         -Djavax.net.ssl.keyStorePassword=key_store_password
```

Tyto vlastnosti nemohou být vlastnostmi objektu ActivationSpec nebo ConnectionFactory a pro aplikační server nelze určit více než jedno úložiště klíčů. Vlastnosti se vztahují na celé prostředí JVM, a mohou proto ovlivnit aplikační server, pokud jiné aplikace spuštěné na aplikačním serveru používají připojení SSL. Aplikační server může tyto vlastnosti také resetovat na různé hodnoty. Další informace o použití zabezpečení SSL s třídami WebSphere MQ pro platformu JMS naleznete v tématu “Použití zabezpečení SSL (Secure Sockets Layer) s třídami produktu WebSphere MQ pro službu JMS” na stránce 875.

Program pro verifikaci instalace (IVT) je dodáván s adaptérem prostředků WebSphere MQ , ale před spuštěním programu je třeba konfigurovat adaptér prostředků. Informace o tom, co je třeba nakonfigurovat, aby bylo možné spustit program IVT, najdete v tématu “Testovací program ověření instalace pro adaptér prostředků WebSphere MQ” na stránce 756.

Protokoly adaptéru prostředků, varování a chybové zprávy používají stejný mechanizmus jako třídy produktu IBM WebSphere MQ pro službu JMS, podrobnosti viz “Protokolování a IBM WebSphere MQ classes for JMS” na stránce 769. Pro WebSphere Application Serverjsou tyto zprávy automaticky

přesměrovány do výstupního protokolu aplikačního serveru. Pro ostatní aplikační servery, jako např. WAS CE a JBoss, se standardně zobrazí soubor s názvem mqjms.log. Chcete-li nakonfigurovat adaptér prostředků tak, aby do standardního výstupního protokolu aplikačního serveru dodatečně protokolovací zprávy protokolovaných zpráv, nastavte následující systémovou vlastnost prostředí JVM pro aplikační server:

```
-Dcom.ibm.msg.client.commonservices.log.outputName=mqjms.log,stdout
```

Podrobnosti o tom, jak nastavit systémovou vlastnost prostředí JVM, najdete v dokumentaci k aplikačnímu serveru.

Konfigurace objektu ResourceAdapter
Objektu ResourceAdapter jsou zapouzdřeny globální vlastnosti adaptéru prostředků produktu WebSphere MQ . Tyto vlastnosti nadefinujte pomocí zařízení pro adaptér prostředků.

Objekt ResourceAdapter má dvě sady vlastností:

- Vlastnosti přidružené k trasování diagnostiky
- Vlastnosti přidružené k fondu připojení spravovanému adaptérem prostředků

Způsob, jakým definujete tyto vlastnosti, závisí na rozhraních pro administraci poskytovaných aplikačním serverem.

Další informace o definování vlastností asociovaných s trasováním diagnostiky naleznete v tématu Trasování adaptéru prostředků produktu IBM WebSphere MQ.

Adaptér prostředků spravuje interní fond připojení pro připojení JMS, která jsou používána k doručování zpráv do objektů MDB. Tabulka 95 na stránce 713 Zobrazí seznam vlastností objektů ResourceAdapter , které jsou přidruženy k fondu připojení.

Tabulka 95. Vlastnosti objektu ResourceAdapter , které jsou přidruženy k fondu připojení

Název vlastnosti	**Typ**	**Výchozí hodnota**	**Popis**
maxConnections	Řetězec	50	Maximální počet připojení ke správci front WebSphere MQ a maximální počet implementovaných objektů MDB.
connectionConcurrency	Řetězec	1	Maximální počet objektů MDB, které mají být sdíleny s připojením JMS. Sdílení připojení není možné a tato vlastnost má vždy hodnotu 1.
Počet reconnectionRetry	Řetězec	5	Maximální počet pokusů provedených adaptérem prostředků k opětovnému připojení ke správci front WebSphere MQ , pokud dojde k selhání připojení.
reconnectionRetryInterval	Řetězec	300 000	Doba v milisekundách, po kterou adaptér prostředků čeká před pokusem o opětovné připojení ke správci front WebSphere MQ .
Počet startupRetryPočet	Řetězec	0	Výchozí počet pokusů o připojení objektu MDB ke spuštění, pokud správce front není spuštěn při spuštění aplikačního serveru.
Interval startupRetryInterval	Řetězec	30 000	Výchozí doba spánku mezi pokusy o spuštění připojení (v milisekundách).

Je-li objekt MDB implementován na aplikačním serveru, vytvoří se nové připojení JMS a zahájí se konverzace se správcem front za předpokladu, že maximální počet připojení určený vlastností maxConnection není překročen. Maximální počet objektů typu message-driven bean je tedy roven

maximálnímu počtu připojení. Pokud počet implementovaných objektů MDB dosáhne tohoto maxima, jakýkoli pokus o implementaci jiného objektu MDB selže. Je-li objekt MDB zastaven, může být jeho připojení používáno jiným objektem MDB.

Obecně platí, že pokud se má implementovat mnoho objektů MDB, je třeba zvýšit hodnotu vlastnosti maxConnections .

Vlastnosti reconnectionRetryCount a reconnectionRetryInterval řídí chování adaptéru prostředků při selhání připojení ke správci front produktu WebSphere MQ , protože došlo k selhání sítě. Dojde-li k selhání připojení, adaptér prostředků pozastaví doručování zpráv do všech objektů MDB dodaných tímto připojením po uplynutí intervalu určeného vlastností intervalu reconnectionRetry. Adaptér prostředků se pak pokusí znovu připojit ke správci front. Pokud se pokus nezdaří, adaptér prostředků provede další pokusy o opětovné připojení v intervalech určených vlastností Interval reconnectionRetry, dokud není dosažena mezní hodnota stanovená vlastností reconnectionRetryCount. Pokud se všechny pokusy nezdaří, je doručení trvale zastaveno, dokud nebudou objekty MDB restartovány ručně.

Obecně objekt ResourceAdapter nevyžaduje žádnou administraci. Chcete-li například povolit diagnostické trasování v systémech UNIX and Linux , můžete nastavit následující vlastnosti:

```
traceEnabled:       true
traceLevel:         10
```

Tyto vlastnosti nemají žádný účinek, pokud adaptér prostředků nebyl spuštěn, což je například případ, kdy jsou aplikace používající prostředky produktu WebSphere MQ spuštěny pouze v kontejneru klienta. V této situaci můžete nastavit vlastnosti pro trasování diagnostiky jako systémové vlastnosti Java Virtual Machine (JVM). Vlastnosti můžete nastavit pomocí příznaku -D u příkazu **java** , jako v následujícím příkladu:

```
java ... -DtraceEnabled=true -DtraceLevel=6
```

Není třeba definovat všechny vlastnosti objektu ResourceAdapter . Všechny vlastnosti ponechané nespecifikované vlastnosti mají své výchozí hodnoty. Ve spravovaném prostředí je lepší nesměšovat dva způsoby určení vlastností. Pokud je směšujete, mají systémové vlastnosti prostředí JVM přednost před vlastnostmi objektu ResourceAdapter .

Konfigurace adaptéru prostředků pro příchozí komunikaci
Chcete-li konfigurovat příchozí komunikaci, definujte vlastnosti jednoho nebo více objektů ActivationSpec .

Vlastnosti objektu ActivationSpec určují způsob, jakým objekt typu message-driven bean (MDB) přijímá zprávy JMS z fronty WebSphere MQ . Transakční chování objektu MDB je definováno v deskriptoru implementace.

Objekt ActivationSpec má dvě sady vlastností:

- Vlastnosti použité k vytvoření připojení JMS ke správci front produktu WebSphere MQ
- Vlastnosti, které se používají k vytvoření spotřebitele připojení JMS, který doručuje zprávy asynchronně při jejich doručení do určené fronty.

Způsob, jakým definujete vlastnosti objektu ActivationSpec , závisí na rozhraních administrace, která poskytuje váš aplikační server.

Tabulka 96 na stránce 715 uvádí vlastnosti objektu ActivationSpec , které se používají k vytvoření připojení JMS ke správci front WebSphere MQ .

Tabulka 96. 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 pomocí příkazu **DISPLAY CONN MQSC/PCF** (kde se pole nazývá **APPLTAG**) nebo v zobrazení IBM WebSphere MQ Průzkumník **Připojení aplikací** (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
brokerSubQueue[1]	Řetězec	• **SYSTEM.JMS.ND.SUBSCRIBER.QUEUE** • Název fronty	Název fronty, ze které spotřebitel přechodných zpráv přijímá zprávy

Tabulka 96. Vlastnosti objektu ActivationSpec , které se používají k vytvoření připojení JMS (pokračování)

Název majetku	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
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 publikování/odběru WebSphere MQ . Nebo můžete použít zprostředkovatele WebSphere MQ Integrator, WebSphere Event Broker, WebSphere Business Integration Event Broker nebo WebSphere Business Integration Message Broker v režimu kompatibility. Tato hodnota je výchozí hodnota, pokud je hodnota TRANSPORT nastavena na BIND nebo CLIENT. • **V2** -Chcete-li použít zprostředkovatele WebSphere MQ Integrator, WebSphere Event Broker, WebSphere Business Integration Event Broker nebo WebSphere Business Integration Message Broker 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 (adresa URL)	Adresa URL, která identifikuje název a umístění souboru obsahujícího tabulku CCDT (Client Channel Definition Table) a určuje, jak lze k souboru přistupovat.
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

Tabulka 96. Vlastnosti objektu ActivationSpec , které se používají k vytvoření připojení JMS (pokračování)			
Název majetku	**Typ**	**Platné hodnoty (výchozí hodnota tučně)**	**Popis**
cleanupLevel[1]	Řetězec	• **Bezpečný** • ŽÁDNÉ • velká • Vynutit • NONDUR	Úroveň vyčištění pro úložiště odběrů založené na zprostředkovateli
clientID	Řetězec	• **null** • Identifikátor klienta	Identifikátor klienta pro připojení
cloneSupport	Řetězec	• **DISABLED** -V daném okamžiku může být spuštěna pouze jedna instance trvalého odběratele tématu. • ENABLED-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
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.
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

Tabulka 96. Vlastnosti objektu ActivationSpec , které se používají k vytvoření připojení JMS (pokračování)

Název majetku	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
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
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 třídy WebSphere MQ pro JMS nebo zprostředkovatel. Výběr zprávy zprostředkovatelem není podporován, pokud má brokerVersion hodnotu 1.

Tabulka 96. Vlastnosti objektu ActivationSpec , které se používají k vytvoření připojení JMS (pokračování)			
Název majetku	**Typ**	**Platné hodnoty (výchozí hodnota tučně)**	**Popis**
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** .
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 třídy WebSphere MQ pro rozhraní Java, MQReceiveExit .	Identifikuje uživatelský program pro příjem kanálu nebo posloupnost uživatelských procedur pro příjem, které mají být spuštěny v posloupnosti.

Tabulka 96. Vlastnosti objektu ActivationSpec , které se používají k vytvoření připojení JMS (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 uživatelským proceduře pro příjem kanálu.
rescanInterval[1]	celé číslo	• **5000** • Libovolné kladné celé číslo	Pokud spotřebitel zpráv v doméně typu point-to-point používá selektor zpráv k výběru zpráv, které chce přijmout, třídy WebSphere MQ pro JMS vyhledává ve frontě WebSphere MQ vhodné zprávy v pořadí určeném atributem *MsgDeliverySequence* fronty. Když třídy WebSphere MQ pro JMS naleznou vhodnou zprávu a doručí ji spotřebiteli, WebSphere MQ třídy pro JMS obnoví hledání další vhodné zprávy z aktuální pozice ve frontě. WebSphere Třídy produktu MQ pro službu JMS pokračují v prohledávání fronty tímto způsobem, dokud nedosáhne konce fronty nebo dokud nevyprší časový interval v milisekundách určený hodnotou této vlastnosti. V každém případě se třídy WebSphere MQ pro JMS vrátí na začátek fronty, aby pokračovaly v hledání, a začne nový časový interval.
securityExit [3]	Řetězec	• **null** • Úplný název třídy, která implementuje třídy WebSphere MQ pro rozhraní 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

Tabulka 96. Vlastnosti objektu ActivationSpec , které se používají k vytvoření připojení JMS (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 třídy WebSphere MQ pro rozhraní 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 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 shodují definice kanálu.
sparseSubodběr[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 adresa 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 uchovávají seznamy odvolaných certifikátů (CRL) pro použití v připojení SSL
SSLCIPHERSUITE	Řetězec	• **null** • Název CipherSuite	CipherSuite , která má být použita pro připojení SSL.
sslFipsVyžadováno [2]	Logická hodnota	• **ne** • ano	Zda musí připojení SSL používat sadu CipherSuite , která je podporována poskytovatelem Java JSSE FIPS (IBMJSSEFIPS) IBM .
SSLPEERNAME	Řetězec	• **null** • Šablona pro rozlišující názvy	V případě připojení SSL se jedná o šablonu, která se používá ke kontrole rozlišujícího názvu v digitálním certifikátu poskytnutém správcem front.

Tabulka 96. Vlastnosti objektu ActivationSpec , které se používají k vytvoření připojení JMS (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 SSL před opětovným vyjednáním tajných klíčů použitých SSL
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 má vlastnost subscriptionStore hodnotu QUEUE.
subscriptionStore[1]	Řetězec	• **BROKER** • MIGRATE • QUEUE	Určuje, kam třídy WebSphere MQ pro platformu JMS ukládají trvalá data o aktivních odběrech.
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.
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 7.0 tříd WebSphere MQ pro platformu JMS. Nemá vliv na aplikaci připojenou ke správci front verze 7.0 , pokud není vlastnost providerVersion nastavena na číslo verze nižší než 7.
2. Důležité informace o použití vlastnosti sslFipsRequired najdete v části "Omezení adaptéru prostředků produktu IBM WebSphere MQ" na stránce 745.
3. Informace o tom, jak nakonfigurovat adaptér prostředků tak, aby mohl vyhledat uživatelskou proceduru, naleznete v části "Konfigurace produktu IBM WebSphere MQ classes for JMS pro použití uživatelských procedur kanálu" na stránce 882.

Tabulka 97 na stránce 723 uvádí vlastnosti objektu ActivationSpec , které se používají k vytvoření spotřebitele připojení JMS.

Tabulka 97. 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.
destinationType	Řetězec	• javax.jms.Queue • javax.jms.Topic	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.
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 pouze v případě, že je aplikace spuštěna na serveru WebSphere Application Server verze 7 nebo novější.
nonASFRollbackPovoleno	Logická hodnota	• **false** -Zpráva se spotřebuje i v případě, že objekt MDB selže. • true-Selhání v objektu MDB způsobí, že se zpráva odvolá do fronty.	Určuje, zda se doručení zpráv nachází v synchronizačním bodu produktu WebSphere MQ , pokud objekt MDB není transakční. Ignoruje se, pokud je objekt MDB transakční nebo pokud je nonASFTimeout nastaven na hodnotu 0.

Tabulka 97. Vlastnosti objektu ActivationSpec , které se používají k vytvoření spotřebitele připojení JMS (pokračování)

Název majetku	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
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.	Zda je objektu MDB povoleno použít dopředné čtení k získání dočasných zpráv z místa určení do interní vyrovnávací paměti před jejich přijetím.
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.
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.

Tabulka 97. Vlastnosti objektu ActivationSpec , které se používají k vytvoření spotřebitele připojení JMS (pokračování)

Název majetku	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
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
useJNDI	Logická hodnota	• **false** -Vlastnost s názvem destination je interpretována jako název fronty nebo tématu produktu WebSphere MQ . • true-Vlastnost s názvem destination je interpretována jako název objektu javax.jms.Queue nebo javax.jms.Topic v oboru názvů JNDI aplikačního serveru.	Určuje způsob interpretace hodnoty vlastnosti s názvem destination

Vlastnosti ActivationSpec s názvem destination a destinationType musí být definovány explicitně. Všechny ostatní vlastnosti jsou volitelné.

Objekt ActivationSpec může mít konfliktní vlastnosti. Můžete například určit vlastnosti zabezpečení SSL pro připojení v režimu vazeb. V tomto případě je chování určeno typem přenosu a doménou systému zpráv, což je buď dvoubodové, 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 i kanál, 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.

Většina vlastností objektu ActivationSpec je ekvivalentní vlastnostem tříd WebSphere MQ pro objekty JMS nebo parametrům tříd WebSphere MQ pro metody JMS. Avšak tři vlastnosti ladění a jedna vlastnost použitelnosti nemají žádné ekvivalenty ve třídách WebSphere MQ pro JMS:

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 97 na stránce 723.

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:

```
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 produktu WebSphere MQ” na stránce 711.

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 jiné 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.

Zpracování příchozích nezpracovatelných zpráv v adaptéru prostředků

Za určitých okolností může být zpráva doručená do objektu typu message-driven bean odvolána do fronty produktu WebSphere 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. Produkt WebSphere MQ můžete nakonfigurovat tak, aby třídy

produktu WebSphere MQ pro platformu JMS automaticky přenesly nezpracovatelnou zprávu do jiné fronty za účelem dalšího zkoumání nebo vyřazení zprávy.

Podrobnosti o tom, jak zpracovat nezpracovatelné zprávy, viz “Zpracování nezpracovatelných zpráv v produktu IBM WebSphere MQ classes for JMS” na stránce 859.

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 systémy UNIX, Linux a Windows

Konfigurace adaptéru prostředků pro odchozí komunikaci

Chcete-li konfigurovat odchozí komunikaci, definujte vlastnosti objektu ConnectionFactory a spravovaného cílového objektu.

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:

```
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 HTTP GET, načte objekt ConnectionFactory a objekt Queue z oboru názvů JNDI a použije tyto objekty k odeslání zprávy do fronty WebSphere MQ . Servlet poté obdrží zprávu, kterou odeslal.

Chcete-li konfigurovat odchozí komunikaci, definujte prostředky 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 fronty JMS nebo objektu tématu JMS.

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 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 doménu typu point-to-point nebo o objekt továrny TopicConnectionpro doménu publikování/odběru.

Tabulka 98 na stránce 728 uvádí vlastnosti objektu ConnectionFactory .

Tabulka 98. 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 WebSphere MQ Průzkumník **Připojení aplikací** (kde se pole nazývá **App name**).
brokerCCSubFronta[1]	Řetězec	• **SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE** • Název fronty	Název fronty, z níž spotřebitel připojení přijímá zprávy dočasného odběru.
brokerControlFronta[1]	Řetězec	• **SYSTEM.BROKER.CONTROL.QUEUE** • Název fronty	Název fronty řízení zprostředkovatele.
brokerPubQueue[1]	Řetězec	• **SYSTEM.BROKER.DEFAULT.STREAM** • Název fronty	Název fronty, kam jsou odesílány publikované zprávy (fronta proudu).
brokerQueueManager[1]	Řetězec	• **"" (prázdný řetězec)** • Název správce front	Název správce front, v němž je zprostředkovatel spuštěn.
brokerSubQueue[1]	Řetězec	• **SYSTEM.JMS.ND.SUBSCRIBER.QUEUE** • Název fronty	Název fronty, ze které spotřebitel přechodných zpráv přijímá zprávy. Další informace viz vlastnost BROKERSUBQ .

Tabulka 98. Vlastnosti objektu ConnectionFactory (pokračování)

Název majetku	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
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 publikování/odběru IBM WebSphere MQ . Nebo chcete-li použít zprostředkovatele produktu IBM WebSphere MQ Integrator, WebSphere Event Broker, WebSphere Business Integration Event Broker nebo WebSphere Business Integration Message Broker v režimu kompatibility. Tato hodnota je výchozí hodnota, pokud je hodnota TRANSPORT nastavena na BIND nebo CLIENT. • **V2** -Chcete-li použít zprostředkovatele produktu IBM WebSphere MQ Integrator, WebSphere Event Broker, WebSphere Business Integration Event Broker nebo WebSphere Business Integration Message Broker 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 (adresa URL)	Adresa 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 virtuálním počítačem Java (JVM).	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.

Tabulka 98. Vlastnosti objektu ConnectionFactory (pokračování)			
Název majetku	**Typ**	**Platné hodnoty (výchozí hodnota tučně)**	**Popis**
cleanupLevel[1]	Řetězec	• **Bezpečný** • ŽÁDNÉ • velká • Vynutit • NONDUR	Úroveň vyčištění pro úložiště odběrů založené na zprostředkovateli.
clientID	Řetězec	• **null** • Identifikátor klienta	Identifikátor klienta pro účely připojení.
cloneSupport	Řetězec	• **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.
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.

Tabulka 98. Vlastnosti objektu ConnectionFactory (pokračování)

Název majetku	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
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 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[1]	Řetězec	• **client** • BROKER	Určuje, zda je výběr zpráv proveden třídami IBM WebSphere MQ pro JMS nebo zprostředkovatelem. 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.

Tabulka 98. Vlastnosti objektu ConnectionFactory (pokračování)

Název majetku	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
pollingInterval[1]	celé číslo	• **5000** • Libovolné kladné celé číslo	Pokud každý modul listener pro zprávy v rámci relace nemá ve frontě žádnou odpovídající zprávu, jedná se o maximální interval v milisekundách, který uplyne předtím, než se každý modul listener pro zprávy znovu pokusí získat zprávu z příslušné fronty. Pokud se často stává, že pro žádný z 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** .
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[1]	celé číslo	• **25** • Kladné celé číslo	Počet zpráv publikovaných vydavatelem, než produkt IBM WebSphere 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 WebSphere MQ classes for Java , MQReceiveExit .	Identifikuje uživatelský program pro příjem kanálu nebo posloupnost uživatelských procedur pro příjem, které mají být spuštěny v posloupnosti.

Tabulka 98. 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[1]	celé číslo	• **5000** • Libovolné kladné celé číslo	Pokud spotřebitel zpráv v doméně typu point-to-point používá selektor zpráv k výběru zpráv, které chce přijmout, třídy WebSphere MQ pro JMS vyhledá ve frontě IBM WebSphere MQ vhodné zprávy v pořadí určeném atributem *MsgDeliverySequence* fronty. Když třídy WebSphere MQ pro JMS naleznou vhodnou zprávu a doručí ji spotřebiteli, WebSphere MQ třídy pro JMS obnoví hledání další vhodné zprávy z aktuální pozice ve frontě. WebSphere Třídy produktu MQ pro službu JMS pokračují v prohledávání fronty tímto způsobem, dokud nedosáhne konce fronty nebo dokud nevyprší časový interval v milisekundách určený hodnotou této vlastnosti. V každém případě se třídy WebSphere MQ pro JMS vrátí na začátek fronty, aby pokračovaly v hledání, a začne nový časový interval.
securityExit[3]	Řetězec	• **null** • Úplný název třídy, která implementuje třídy WebSphere MQ pro rozhraní 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 98. 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 třídy WebSphere MQ pro rozhraní 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 dalšími připojeními JMS nejvyšší úrovně ze stejného procesu ke stejnému správci front, pokud se definice kanálu shodují.
sparseSubodběr[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 adresa 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 SSL.
SSLCIPHERSUITE	Řetězec	• **null** • Název CipherSuite	CipherSuite , která má být použita pro připojení SSL.
sslFipsVyžadováno [2]	Logická hodnota	• **ne** • ano	Zda musí připojení SSL používat sadu CipherSuite , kterou podporuje poskytovatel IBM Java JSSE FIPS (IBMJSSEFIPS).
SSLPEERNAME	Řetězec	• **null** • Šablona pro rozlišující názvy	Pro připojení SSL se jedná o šablonu, která se používá ke kontrole rozlišujícího názvu v digitálním certifikátu poskytnutém správcem front.

Tabulka 98. 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 SSL před opětovným vyjednáním tajných klíčů použitých SSL.
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[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 má SUBSTORE hodnotu QUEUE.
subscriptionStore[1]	Řetězec	• **BROKER** • MIGRATE • QUEUE	Určuje, kde třídy WebSphere MQ pro platformu JMS ukládají 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 má příchozí zpráva záhlaví MQRFH2 .

Tabulka 98. 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 vytvářeny dočasné fronty JMS. Použít SYSTEM.DEFAULT.MODEL.QUEUE , pokud platí obě následující podmínky: • 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 těchto 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 YES a atributem DEFSOPT nastaveným na 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 98. 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 WebSphere MQ . Pravidla pro vytváření předpony jsou stejná jako pravidla pro vytváření obsahu pole *DynamicQName* v deskriptoru objektu IBM WebSphere MQ , struktura MQOD, ale poslední neprázdný znak musí být hvězdička (*). Je-li hodnotou vlastnosti prázdný řetězec, třídy WebSphere MQ pro platformu JMS používají hodnotu AMQ.* při vytváření dynamické fronty.	Předpona, která se používá k vytvoření názvu dynamické fronty IBM WebSphere MQ .
TEMPTOPICPREFIX	Řetězec	Jakýkoli nenulový řetězec sestávající pouze z platných znaků pro řetězec tématu IBM WebSphere MQ	Při vytváření dočasných témat vygeneruje platforma JMS řetězec tématu ve tvaru "TEMP/*TEMPTOPICPREFIX*/*unique_id*", nebo pokud je tato vlastnost ponechána s výchozí hodnotou, pouze "TEMP/*unique_id*". Zadání neprázdné předpony TEMPTOPICPREFIX umožňuje definovat specifické modelové fronty pro vytváření spravovaných front pro odběratele dočasných témat vytvořených v rámci tohoto připojení.
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.
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.

Tabulka 98. Vlastnosti objektu ConnectionFactory (pokračování)

Název majetku	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis

Notes:

1. Tuto vlastnost lze použít s verzí 7.0 produktu IBM WebSphere MQ classes for JMS , ale nemá vliv na aplikaci připojenou ke správci front verze 7.0 , pokud není vlastnost providerVersion nastavena na číslo verze nižší než 7.
2. Důležité informace o použití vlastnosti sslFipsRequired najdete v části "Omezení adaptéru prostředků produktu IBM WebSphere MQ" na stránce 745.
3. Informace o tom, jak nakonfigurovat adaptér prostředků tak, aby mohl vyhledat uživatelskou proceduru, naleznete v části "Konfigurace produktu IBM WebSphere MQ classes for JMS pro použití uživatelských procedur kanálu" na stránce 882.

Následující příklad ukazuje typickou sadu vlastností objektu ConnectionFactory :

```
channel:        SYSTEM.DEF.SVRCONN
hostName:       192.168.0.42
port:           1414
queueManager:   ExampleQM
transportType:  CLIENT
```

Tabulka 99 na stránce 738 obsahuje seznam vlastností, které jsou společné pro objekt fronty a objekt tématu.

Tabulka 99. 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 virtuálním počítačem Java (JVM).	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.

Tabulka 99. Vlastnosti společné pro objekt fronty a objekt tématu (pokračování)

Název majetku	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
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.
trvání, perzistence	Řetězec	• **APP** -Trvání zprávy je určeno producentem zpráv. • QDEF-Trvání zprávy je určeno atributem *DefPersistence* fronty WebSphere MQ . • PERS-Zpráva je trvalá. • Zpráva typu NON-A je dočasná. • HIGH-Trvání zprávy je určeno atributem *NonPersistentMessageClass* fronty WebSphere MQ podle vysvětlení v části “Trvalé zprávy JMS” na stránce 874.	Perzistence zprávy odeslané do místa určení.
priorita	Řetězec	• **APP** -Priorita zprávy je určena producentem zpráv. • QDEF-Priorita zprávy je určena atributem *DefPriority* fronty IBM WebSphere 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.

Tabulka 99. Vlastnosti společné pro objekt fronty a objekt tématu (pokračování)			
Název majetku	**Typ**	**Platné hodnoty (výchozí hodnota tučně)**	**Popis**
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` .
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 než JMS IBM WebSphere 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 100 na stránce 740 jsou uvedeny vlastnosti specifické pro objekt fronty.

Tabulka 100. 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 WebSphere MQ .
Název baseQueue	Řetězec	• **"" (prázdný řetězec)** • Název fronty	Název základní fronty IBM WebSphere MQ .

Tabulka 101 na stránce 740 uvádí vlastnosti, které jsou specifické pro objekt Topic.

Tabulka 101. 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[1]	Řetězec	• **SYSTEM.JMS.D.CC.SUBSCRIBER.QUEUE** • Název fronty	Název fronty, ze které spotřebitel připojení přijímá zprávy trvalého odběru.

Tabulka 101. Vlastnosti specifické pro objekt Topic (pokračování)

Název majetku	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
brokerDurSubQueue[1]	Řetězec	• **SYSTEM.JMS.D.SUBSCRIBER.QUEUE** • Název fronty	Název fronty, ze které trvalý odběratel tématu přijímá zprávy. Další informace naleznete ve vlastnosti BROKEDURRSUBQ v dokumentaci k Průzkumníku produktu WebSphere MQ .
brokerPubQueue[1]	Řetězec	• **Nenastaveno** • Název fronty	Název fronty, kam jsou odesílány publikované zprávy (fronta proudu). Hodnota této vlastnosti přepíše hodnotu vlastnosti fronty brokerPubobjektu ConnectionFactory . Pokud však nenastavíte hodnotu této vlastnosti, bude použita hodnota vlastnosti fronty brokerPubobjektu ConnectionFactory .
brokerPubQueueManager[1]	Řetězec	• **"" (prázdný řetězec)** • Název správce front	Název správce front, který vlastní frontu, do které jsou odesílány zprávy publikované v tématu.
brokerVersion[1]	Řetězec	• **Nenastaveno** • 1 • 2	Verze používaného zprostředkovatele. Hodnota této vlastnosti přepíše hodnotu vlastnosti brokerVersion objektu ConnectionFactory . Pokud však nenastavíte hodnotu této vlastnosti, použije se hodnota vlastnosti brokerVersion objektu ConnectionFactory .

Poznámka:

1. Tuto vlastnost lze použít s verzí 7.0 produktu IBM WebSphere MQ classes for JMS , ale nemá vliv na aplikaci připojenou ke správci front verze 7.0 , pokud není vlastnost providerVersion objektu ConnectionFactory nastavena na číslo verze nižší než 7.

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 za běhu jsou v klientu MQI použity pouze specifikace CipherSpecs s certifikací FIPS.

Související odkazy

Standard FIPS (Federal Information Processing Standards) pro systémy UNIX, Linux a Windows

V 7.5.0.9 *Konfigurace vlastnosti Vyhovující klienta targetClientpro specifikaci aktivace*

Můžete nakonfigurovat vlastnost **targetClientMatching** pro specifikaci aktivace tak, aby bylo záhlaví MQRFH2 zahrnuto ve zprávách odpovědi, když zprávy požadavku neobsahují záhlaví MQRFH2 . To znamená, že všechny vlastnosti zprávy, které aplikace definuje na zprávě s odpovědí, jsou zahrnuty při odeslání zprávy.

Informace o této úloze

Pokud aplikace objektu typu message-driven bean (MDB) spotřebovává zprávy, které neobsahují záhlaví MQRFH2 , prostřednictvím specifikace aktivace adaptéru prostředků JCA IBM WebSphere MQ a následně odesílá zprávy odpovědi na místo určení rozhraní JMS vytvořené z pole JMSReplyTo zprávy požadavku, zprávy odpovědi musí obsahovat záhlaví MQRFH2 , i když zprávy požadavku nepracují, jinak budou ztraceny všechny vlastnosti zprávy, které aplikace definovala ve zprávě odpovědi.

Vlastnost **targetClientMatching** definuje, zda zpráva odpovědi odeslaná do fronty označené v poli záhlaví JMSReplyTo příchozí zprávy má záhlaví MQRFH2 pouze v případě, že má příchozí zpráva záhlaví MQRFH2 . Tuto vlastnost můžete nakonfigurovat pro specifikaci aktivace v obojích WebSphere Application Server traditional i WebSphere Application Server Liberty.

Nastavíte-li hodnotu vlastnosti **targetClientMatching** na hodnotu `false`, záhlaví MQRFH2 lze zahrnout do zprávy odpovědi odeslané do cíle rozhraní JMS vytvořeného z záhlaví JMSReplyTo příchozí zprávy požadavku, která neobsahuje MQRFH2. Důvodem je skutečnost, že vlastnost **targetClient** na cíli JMS je nastavena na hodnotu `0`, což znamená, že zprávy obsahují záhlaví MQRFH2 . Přítomnost záhlaví MQRFH2 v odchozí zprávě umožňuje ukládání vlastností zpráv definovaných uživatelem ve zprávě při odeslání do fronty produktu IBM WebSphere MQ .

Je-li vlastnost **targetClientMatching** nastavena na hodnotu `true` a zpráva požadavku neobsahuje záhlaví MQRFH2 , záhlaví MQRFH2 nebude zahrnuto do zprávy odpovědi.

Procedura

- V produktu WebSphere Application Server traditional použijte konzolu pro správu k definování vlastnosti **targetClientMatching** jako přizpůsobené vlastnosti ve specifikaci aktivace IBM WebSphere MQ :
  a) V navigačním podokně klepněte na **Prostředky-> JMS-> Specifikace aktivace**.
  b) Vyberte název specifikace aktivace, kterou chcete zobrazit nebo změnit.
  c) Klepněte na volbu **Přizpůsobené vlastnosti-> Nový** a poté zadejte podrobnosti nové přizpůsobené vlastnosti.

  Nastavte název vlastnosti na `targetClientMatching`, typ na `java.lang.Boolean` a hodnotu na `false`.
- V produktu WebSphere Application Server Liberty zadejte vlastnost **targetClientMatching** v 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 848

Místo načítání cílů jako spravovaných objektů z oboru názvů rozhraní JNDI (Java Naming and Directory Interface) může aplikace platformy JMS použít relaci k dynamickému vytváření cílů za běhu. Aplikace

může použít identifikátor URI (Uniform Resource Identifier) k identifikaci fronty nebo tématu produktu WebSphere MQ a volitelně k určení jedné či více vlastností objektu Fronta nebo Téma.

"Konfigurace adaptéru prostředků pro odchozí komunikaci" na stránce 727
Chcete-li konfigurovat odchozí komunikaci, definujte vlastnosti objektu ConnectionFactory a spravovaného cílového objektu.

Režim ASF a non-ASF
Režim ASF (Application Server Facilities-ASF) je výchozí metoda, kterou služba modulu listener pro zprávy v produktu WebSphere Application Server zpracovává zprávy.

Služba listener pro zprávy má dva režimy provozu, zařízení ASF (Application Server Facilities) a non-ASF (non-Application Server Facilities):

- Režim ASF poskytuje souběžnou práci a podporu transakcí pro aplikace. U objektů typu message-driven bean pro publikování/odběr poskytuje režim ASF lepší propustnost a souběžnost, protože v režimu non-ASF je modul listener pouze jedním vláknem.
- Režim non-ASF je určen hlavně pro použití s poskytovateli systému zpráv jiného dodavatele, kteří nepodporují standard JMS ASF, který je volitelným rozšířením specifikace JMS. Režim non-ASF je také transakční, ale protože délka cesty je kratší než pro režim ASF, obvykle poskytuje lepší výkon.

Chcete-li povolit režim non-ASF pro všechny moduly listener objektů typu message-driven bean na aplikačním serveru, nastavte tuto vlastnost na hodnotu bez nuly.

Poznámka:

Režim non-ASF nelze vybrat na systémech z/OS , takže v tomto případě nesmíte nastavit jinou než nulovou hodnotu této vlastnosti.

Zpracování zpráv v režimu ASF

V režimu ASF jsou relace serveru a podprocesy přiděleny pouze pro práci, je-li zjištěna zpráva, která je vhodná pro objekt MDB (Message-driven bean). Počet podprocesů, které může objekt MDB zpracovat souběžně, je určen hodnotou vlastnosti **Maximum Sessions** pro port modulu listener nebo specifikaci aktivace.

Zpracování zpráv v režimu non-ASF

Podprocesy režimu non-ASF jsou aktivní od chvíle, kdy je spuštěn port modulu listener nebo specifikace aktivace. Počet aktivních podprocesů je určen hodnotou zadanou pro vlastnost **Maximum Sessions** . Počet podprocesů určených ve vlastnosti **Maximum Sessions** je aktivní bez ohledu na počet zpráv, které jsou k dispozici pro zpracování. Každý aktivní podproces je individuálním fyzickým síťovým připojením.

Produkt IBM WebSphere MQ verze 7.0 nebo novější vám umožňuje mít až deset podprocesů sdílení jednoho fyzického síťového připojení.

Související pojmy
TřídyIBM WebSphere MQ pro zařízení aplikačního serveru JMS
Toto téma popisuje, jak produkt WebSphere MQ Classes for JMS implementuje třídu ConnectionConsumer a rozšířenou funkčnost ve třídě relace. Shrnuje také souhrn funkce fondu relací serveru.

Související úlohy
Konfigurace aktivačních specifikací pro non-ASF režim
Specifikace aktivace jsou standardizovaným způsobem správy a konfigurace vztahu mezi objektem typu message-driven bean (MDB) spuštěným v produktu WebSphere Application Server a místem určení v produktu IBM WebSphere MQ. Tato úloha vysvětluje, jak nakonfigurovat produkt WebSphere Application Server pro použití režimu non-ASF pro zpracování zpráv.

Související informace
Zpracování zpráv v režimu ASF a v režimu non-ASF

Konfigurace specifikací aktivace pro režim non-ASF
Specifikace aktivace jsou standardizovaným způsobem správy a konfigurace vztahu mezi objektem typu message-driven bean (MDB) spuštěným v produktu WebSphere Application Server a místem určení v produktu IBM WebSphere MQ. Tato úloha vysvětluje, jak nakonfigurovat produkt WebSphere Application Server pro použití režimu non-ASF pro zpracování zpráv.

Než začnete

Způsob, jakým definujete vlastnosti specifikace aktivace, závisí na rozhraních pro administraci poskytnutých aplikačním serverem. Tato úloha předpokládá, že používáte produkt WebSphere Application Server verze 7 nebo novější jako aplikační server a produkt IBM WebSphere MQ jako poskytovatele systému zpráv.

Poznámka:

Režim non-ASF nelze vybrat na systémech z/OS .

Informace o této úloze

Vlastnosti specifikace aktivace určují, jak objekt MDB (Message drive bean) přijímá zprávy platformy JMS z fronty produktu IBM WebSphere MQ . Chcete-li nakonfigurovat režim non-ASF, definujte vlastnosti jedné nebo více specifikací aktivace.

Existuje několik konfigurací produktu IBM WebSphere MQ , které můžete použít v režimu non-ASF. S následujícími konfiguracemi používá každý podproces samostatné fyzické připojení k síti:

- Správce front IBM WebSphere MQ verze 7.x s použitím továrny připojení, která má vlastnost verze poskytovatele nastavenou na hodnotu 6.
- Správce front produktu IBM WebSphere MQ verze 7.x s použitím faktorie připojení, která má nastavovanou vlastnost verze poskytovatele na hodnotu 7 nebo nespecifikované, připojení prostřednictvím kanálu produktu IBM WebSphere MQ s parametrem sdílení **SHARECNV** (sdílení konverzací) nastaveným na hodnotu 0.

Chcete-li konfigurovat non-ASF, nastavte vlastnost ActivationSpec **NON.ASF.RECEIVE.TIMEOUT** na kladné celé číslo, které indikuje, že se používá doručení non-ASF. Hodnota je čas (v milisekundách), po který požadavek na získání čeká na zprávy, které možná ještě nedorazily (get with wait call). Předvolená hodnota 0 označuje, že se použije funkce ASF doručení. Další podrobnosti viz **Přizpůsobené vlastnosti služby listener pro zprávy**.

Tento parametr je platný pouze v případě, že je aplikace spuštěna na serveru WebSphere Application Server verze 7 nebo novější.

Postup

1. Spusťte administrativní konzolu serveru WebSphere Application Server .
2. Zobrazit stránku s nastavením služby modulu listener:
   a) V navigačním podokně vyberte volbu **Servery > Typy serverů > Aplikační servery WebSphere**.
   b) V podokně obsahu klepněte na název aplikačního serveru.
   c) V části **Komunikace**klepněte na volbu **Systém zpráv > Služba listener pro zprávy**.
3. Nastavte přizpůsobenou vlastnost **NON.ASF.RECEIVE.TIMEOUT** jako přizpůsobené vlastnosti služby listener pro zprávy.
   a) Klepněte na volbu **Přizpůsobené vlastnosti**.
   b) Klepněte na volbu **Nový**.
   c) Do pole **Název** zadejte název vlastnosti, která má být nastavena na hodnotu **NON.ASF.RECEIVE.TIMEOUT**.
   d) Zadejte požadovanou hodnotu do pole **Hodnota** .
   e) Klepněte na tlačítko **OK**.

4. Uložte své změny do hlavní konfigurace.
5. Chcete-li aktivovat změněnou konfiguraci, zastavte a poté restartujte aplikační server.

Výsledky

Nakonfigurovali jste vlastnosti služby listener pro zprávy pro produkt WebSphere Application Server pro použití režimu non-ASF.

Poznámka: Při použití režimu non-ASF je třeba zajistit, aby bylo pro zpracování dokončeno před dosažením celkového časového limitu doby trvání transakce dostatečné množství času, aby nedošlo k nechtěnému vypršení časového limitu transakcí. Další podrobnosti viz **NON.ASF.RECEIVE.TIMEOUT** v dokumentaci produktu WebSphere Application Server .

Související pojmy

"Režim ASF a non-ASF" na stránce 743
Režim ASF (Application Server Facilities-ASF) je výchozí metoda, kterou služba modulu listener pro zprávy v produktu WebSphere Application Server zpracovává zprávy.

Konfigurace adaptéru prostředků pro příchozí komunikaci
Chcete-li konfigurovat příchozí komunikaci, definujte vlastnosti jednoho nebo více objektů ActivationSpec .

Související informace

Objekty typu message-driven bean

Služba listener pro zprávy

Zpracování zpráv v režimu ASF a v režimu non-ASF

Jak jsou zprávy zpracovány v non-ASF režimu

Omezení adaptéru prostředků produktu IBM WebSphere MQ

Používáte-li adaptér prostředků produktu IBM WebSphere MQ , některé funkce produktu IBM WebSphere MQ jsou nedostupné nebo omezené.

Adaptér prostředku IBM WebSphere MQ má následující omezení:

- Adaptér prostředků produktu IBM WebSphere MQ je podporován na všech platformách IBM WebSphere MQ s výjimkou systému z/OS.
- Adaptér prostředků produktu IBM WebSphere MQ nepodporuje v reálném čase připojení ke zprostředkovateli. Podporuje pouze připojení ke správci front IBM WebSphere MQ v režimu klienta nebo vazby.
- Adaptér prostředků IBM WebSphere MQ nepodporuje ukončovací programy kanálu, které jsou napsány v jiných jazycích než Java.
- Když je aplikační server spuštěn, hodnota vlastnosti Required sslFipsmusí mít hodnotu true pro všechny prostředky JCA nebo false pro všechny prostředky JCA. Tento požadavek se týká i v případě, že prostředky JCA nejsou souběžně používány. Pokud požadovaná vlastnost sslFipsmá odlišné hodnoty pro různé prostředky JCA, IBM WebSphere MQ vydává kód příčiny MQRC_UNSUPPORTED_CIPHER_SUITE, i když se nepoužívá připojení SSL.
- Pro aplikační server nelze určit více než jedno úložiště klíčů. Pokud jsou připojení prováděna ve více než jednom správci front, musí všechna připojení používat stejné úložiště klíčů. Toto omezení se nevztahuje na produkt WebSphere Application Server.
- Pokud používáte tabulku CCDT (Client Channel Definition CCDT) s více než jednou vhodnou definicí kanálu pro připojení klienta, může v případě selhání adaptéru prostředku vybrat jinou definici kanálu a tudíž jiného správce front z tabulky CCDT, která by způsobila problémy při obnově transakcí. Adaptér prostředků neprovede žádnou akci, aby zabránil použití takové konfigurace, a je vaší zodpovědností vyhnout se konfiguracím, které by mohly způsobit problémy při obnově transakcí.
- Funkce opakování připojení zavedená v produktu IBM WebSphere MQ Version 7.0.1 není podporována pro odchozí připojení při spuštění v kontejneru JEE (EJB/Servlet). Nový pokus o připojení není podporován pro odchozí rozhraní JMS pouze v případě, že je adaptér používán v kontextu kontejneru JEE, a to bez ohledu na konfiguraci transakce nebo pro použití bez transakcí.

Související úlohy

Určení, že pro běhové prostředí klienta MQI je použit pouze certifikovaný standard FIPS CipherSpecs

Související odkazy

Federální standardy zpracování informací (FIPS) pro UNIX, Linux a Windows

Poinstalační nastavení pro třídy produktu WebSphere MQ pro aplikace JMS

Toto téma informuje o tom, jaké oprávnění potřebují třídy WebSphere MQ pro aplikace JMS, aby bylo možné získat přístup k prostředkům správce front. Také uvádí režimy připojení a popisuje, jak nakonfigurovat správce front, aby se aplikace mohly připojit v režimu klienta.

Nezapomeňte zkontrolovat soubor Readme produktu WebSphere MQ . Může obsahovat informace, které nahrazují informace v tomto tématu.

Objekty používané službou JMS, které vyžadují autorizaci pro neprivilegované uživatele

Neprivilegovaní uživatelé potřebují udělit autorizaci pro přístup k frontám používaným platformou JMS. Každá aplikace JMS potřebuje autorizaci ke správci front, se kterým pracuje.

Podrobnosti o řízení přístupu v produktu IBM WebSphere MQnaleznete v tématu Nastavení zabezpečení v systému Windows, UNIX and Linux .

Třídy WebSphere MQ pro aplikace JMS vyžadují ke správci front oprávnění `connect` a `inq` . Můžete nastavit příslušná oprávnění pomocí obslužného příkazu **`setmqaut`** , například:

```
setmqaut -m QM1 -t qmgr -g jmsappsgroup +connect +inq
```

Pro doménu typu point-to-point jsou vyžadovány následující oprávnění:

- Fronty, které používají objekty MessageProducer , potřebují oprávnění `put` .
- Fronty, které používají objekty MessageConsumer a QueueBrowser , potřebují oprávnění `get`, `inq`a `browse` .
- Metoda QueueSession.createTemporaryQueue () potřebuje přístup k modelové frontě určené vlastností TEMPMODEL objektu továrny QueueConnection. Ve výchozím nastavení je tato modelová fronta SYSTEM.TEMP.MODEL.QUEUE.

Jsou-li některé z těchto front alias fronty, jejich cílové fronty vyžadují dotazové oprávnění. Je-li cílová fronta frontou klastru, vyžaduje také oprávnění k procházení.

Pro doménu publikování/odběru se používají následující fronty, pokud se třídy WebSphere MQ pro JMS připojují ke správci front produktu IBM WebSphere MQ v režimu migrace poskytovatele systému zpráv IBM WebSphere MQ :

- SYSTEM.JMS.ADMIN.QUEUE
- SYSTEM.JMS.REPORT.QUEUE
- SYSTEM.JMS.MODEL.QUEUE
- SYSTEM.JMS.PS.STATUS.QUEUE
- SYSTEM.JMS.ND.SUBSCRIBER.QUEUE
- SYSTEM.JMS.D.SUBSCRIBER.QUEUE
- SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE
- SYSTEM.JMS.D.CC.SUBSCRIBER.QUEUE
- SYSTEM.BROKER.CONTROL.QUEUE

Další informace o režimu migrace poskytovatele systému zpráv produktu IBM WebSphere MQ naleznete v tématu Kdy použít produkt **`PROVIDERVERSION`** .

Kromě toho, pokud se třídy WebSphere MQ pro JMS připojují ke správci front v tomto režimu, každá aplikace, která publikuje zprávy, potřebuje přístup k frontě proudu určené továrnou TopicConnectionnebo objektem tématu. Při výchozím nastavení je tato fronta SYSTEM.BROKER.DEFAULT.STREAM.

Používáte-li produkt ConnectionConsumer, adaptér prostředků IBM WebSphere MQ nebo poskytovatele systému zpráv produktu WebSphere Application Server IBM WebSphere MQ , může být zapotřebí další autorizace.

Fronty, které mají být načteny ConnectionConsumer , musí mít oprávnění `get` , `inqa` `browse` . Fronta nedoručených zpráv systému a fronta zpráv typu backend nebo fronta zpráv používaná serverem ConnectionConsumer musí mít oprávnění `put` a `passall` .

Pokud aplikace používá normální režim poskytovatele systému zpráv WebSphere MQ k provedení systému zpráv publikování/odběru, aplikace využívá integrované funkce publikování/odběru poskytované správcem front. Informace o zabezpečení témat a front, které se používají, najdete v tématu Zabezpečení publikování a odběru .

Režimy připojení pro třídy WebSphere MQ pro JMS

Třídy produktu WebSphere MQ pro aplikaci JMS se mohou připojit ke správci front v režimu klienta nebo vazby. V režimu klienta se třídy produktu WebSphere MQ pro rozhraní JMS připojují ke správci front prostřednictvím protokolu TCP/IP. V režimu vazeb se třídy WebSphere MQ pro rozhraní JMS připojují přímo ke správci front s použitím rozhraní JNI (Java Native Interface).

Aplikace spouštěná v produktu WebSphere Application Server v systému z/OS se může připojit ke správci front buď v rámci vazeb, nebo v režimu klienta, ale aplikace běžící v libovolném 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 buď v rámci vazeb, nebo v režimu klienta.

Můžete použít aktuální nebo starší podporovanou verzi tříd WebSphere MQ pro službu JMS s aktuálním správcem front a můžete použít aktuální nebo dřívější podporovanou verzi správce front s aktuální verzí tříd WebSphere MQ pro JMS. Pokud směšujete různé verze, funkce je omezena na úroveň předchozí verze.

Následující části popisují podrobněji každý režim připojení.

Režim klienta

Chcete-li se připojit ke správci front v režimu klienta, třídy produktu WebSphere MQ pro aplikaci JMS mohou běžet na stejném systému, v němž je spuštěn správce front, nebo na jiném systému. V každém případě se třídy produktu WebSphere MQ pro rozhraní JMS připojují ke správci front prostřednictvím protokolu TCP/IP.

Režim vazeb

Chcete-li se připojit ke správci front v režimu vazeb, musí být třídy produktu WebSphere MQ pro aplikaci JMS spouštěny ve stejném systému, v němž je spuštěn správce front.

Třídy WebSphere MQ pro rozhraní JMS se přímo připojí ke správci front pomocí rozhraní JNI (Java Native Interface). Chcete-li použít přenos vazeb, musí být třídy WebSphere MQ pro platformu JMS spouštěny v prostředí, které má přístup ke knihovnám nativního rozhraní prostředí Java produktu WebSphere MQ . Další informace naleznete v tématu “Konfigurace knihoven JNI (Java Native Interface)” na stránce 701 .

Třídy WebSphere MQ pro platformu JMS podporují následující hodnoty pro *ConnectOption* :

- VAZBA MQCNO_FASTPATH_BINDING
- VAZBA MQCNO_STANDARD_BINDING
- CQCNO_SHARED_BINDING
- VAZBA MQCNO_ISOLATED_BINDING
- MQCNO_SERIALIZE_CONN_TAG_QSG
- MQCNO_RESTRICT_CONN_TAG_QSG
- MQCNO_SERIALIZE_CONN_TAG_Q_MGR
- MQCNO_RESTRICT_CONN_TAG_Q_MGR

Chcete-li změnit volby připojení používané třídami produktu WebSphere MQ pro službu JMS, upravte vlastnost továrny připojení CONNOPT.

Další informace o možnostech připojení naleznete v tématu “Připojení ke správci front pomocí volání MQCONNX” na stránce 201 .

Chcete-li použít přenos vazeb, musí prostředí Java Runtime Environment, které se používá, podporovat identifikátor kódované znakové sady (CCSID) správce front, ke kterému se připojuje třídy WebSphere MQ pro produkt JMS .

Podrobnosti o tom, jak určit, jaké CCSID jsou podporovány běhovým prostředím Java, lze nalézt v WebSphere MQ FDC with Probe id 21 generovaných při použití tříd WebSphere MQ V7 pro jazyk Java nebo WebSphere MQ V7 pro platformu JMS .

Vazby, pak režim klienta

Toto nastavení je výchozí. Při připojování ke správci front v tomto režimu se třídy produktu WebSphere MQ pro aplikaci JMS pokusí o připojení v režimu vazeb, který vyžaduje, aby správce front byl umístěn ve stejném počítači jako aplikace. Pokud bude připojení neúspěšné, aplikace se pak pokusí připojit v režimu klienta a povolit, aby správce front byl umístěn buď lokálně na stejném počítači jako aplikace, nebo vzdáleně.

Konfigurace správce front tak, aby se třídy produktu WebSphere MQ pro aplikace platformy JMS mohly připojit v režimu klienta

Chcete-li nakonfigurovat správce front tak, aby se třídy WebSphere MQ pro aplikace platformy JMS mohly připojit v režimu klienta, je třeba vytvořit definici kanálu připojení serveru a spustit modul listener.

Na systému z/OSmusí být nainstalována funkce Attachment Client Attachment.

Vytvoření definice kanálu připojení serveru

Na všech platformách můžete pomocí příkazu MQSC DEFINE CHANNEL vytvořit definici kanálu připojení serveru. Prohlédněte si následující příklad:

```
DEFINE CHANNEL(JAVA.CHANNEL) CHLTYPE(SVRCONN) TRPTYPE(TCP)
```

Na systému IBM imůžete 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.

Definici kanálu připojení k serveru lze také vytvořit pomocí Průzkumníka IBM WebSphere MQ , který je spuštěn v systému Linux a Windows, nebo na panelech operací a ovládacích panelů v systému z/OS.

Název kanálu (JAVA.CHANNEL v předchozích příkladech) musí být stejný jako název kanálu zadaný vlastností CHANNEL továrny připojení, kterou vaše aplikace používá pro připojení ke správci front. Výchozí hodnota vlastnosti CHANNEL je SYSTEM.DEF.SVRCONN.

Spuštění modulu listener

Je třeba spustit modul listener pro správce front, pokud ještě jeden není spuštěn.

Na všech platformách můžete ke spuštění modulu listener spustit příkaz MQSC START LISTENER, ale s výjimkou systému z/OSje třeba nejprve vytvořit objekt modulu listener pomocí příkazu MQSC DEFINE LISTENER. Prohlédněte si následující příklad:

```
DEFINE LISTENER(LISTENER.TCP) TRPTYPE(TCP) PORT(1414)
START LISTENER(LISTENER.TCP)
```

V systémech UNIX, Linuxa Windows můžete modul listener spustit také pomocí příkazu **runmqlsr** modulu listener, jak je uveden v následujícím příkladu:

```
runmqlsr -t tcp -p 1414 -m QMgrName
```

V tomto příkazu je *QMgrName* název vašeho správce front.

Modul listener můžete také spustit pomocí programu Průzkumník produktu WebSphere MQ , který je spuštěn v produktu Linux a v produktu Windows, nebo na panelech operací a ovládacích panelů v systému z/OS.

Číslo portu, na kterém listener naslouchá, musí být stejné jako číslo portu zadané vlastností PORT továrny připojení, kterou vaše aplikace používá pro připojení ke správci front. Výchozí hodnota vlastnosti PORT je 1414.

Ověřovací test dvoubodové instalace pro třídy WebSphere MQ pro JMS

Program IVT (point-to-point installation verification test) je dodáván s třídami WebSphere MQ pro platformu JMS. 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 viz “Typy objektů JMS” na stránce 906.

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.

Je poskytnut skript pro spuštění programu IVT. Skript se nazývá IVTRun na systémech UNIX and Linux a IVTRun.bat na systému Windowsa nachází se v podadresáři `bin` tříd WebSphere MQ pro instalační adresář JMS.

Chcete-li spustit test v režimu vazeb, zadejte následující příkaz:

```
IVTRun -nojndi [-m qmgr] [-v providerVersion] [-t]
```

Chcete-li spustit test v režimu klienta, nejprve nastavte správce front podle popisu v části “Příprava a spuštění ukázkových programů” na stránce 104. 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]
```

V systémech z/OS není k dispozici žádný ekvivalentní skript, ale IVT můžete spustit v režimu vazeb přímým vyvoláním třídy Java pomocí následujícího příkazu:

```
java com.ibm.mq.jms.MQJMSIVT -nojndi [-m qmgr] [-v providerVersion] [-t]
```

Cesta ke třídě musí obsahovat soubor com.ibm.mqjms.jar.

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, naleznete v popisu vlastnosti PROVIDERVERSION v části Vlastnosti IBM WebSphere 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 zapnuto. Ve výchozím nastavení je trasování vypnuto.

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: WebSphere MQ Client for Java
    JMSXDeliveryCount: 1
    JMS_IBM_PutDate: 20070815
    JMS_IBM_PutTime: 09310400
    JMS_IBM_Format: MQSTR
    JMS_IBM_MsgType: 8
A simple text message from the MQJMSIVT
Reply string equals original string
Closing QueueReceiver
Closing QueueSender
Closing Session
Closing Connection
IVT completed OK
IVT finished
```

Ověřovací test dvoubodové instalace 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 WebSphere MQ JMS tak, aby mohl používat adresářovou službu k ukládání spravovaných objektů. Další informace o těchto předpokladech viz “Nezbytné předpoklady pro třídy WebSphere MQ pro JMS” na stránce 694. Informace o konfiguraci nástroje pro administraci rozhraní JMS WebSphere MQ naleznete v tématu “Konfigurace nástroje pro administraci služby JMS” na stránce 903.

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 UNIX and Linux a IVTSetup.bat na systému Windowsa nachází se v podadresáři `bin` tříd WebSphere MQ pro instalační adresář JMS. Chcete-li spustit skript, zadejte následující příkaz:

```
IVTSetup
```

Skript vyvolá nástroj administrace WebSphere MQ JMS pro vytvoření spravovaných objektů.

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, musíte použít nástroj pro administraci JMS WebSphere MQ nebo WebSphere MQ Explorer a změnit příslušné vlastnosti objektu továrny MQQueueConnection. Informace o použití nástroje pro administraci produktu WebSphere MQ JMS naleznete v části “Použití nástroje pro administraci produktu WebSphere MQ JMS” na stránce 902. Informace o použití produktu WebSphere MQ Explorer naleznete v nápovědě k produktu WebSphere 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*"
Adresa URL (Uniform Resource Locator) adresářové služby. Adresa URL může mít jeden z následujících formátů:

- `ldap://`*`hostname`*`/`*`contextName`*, pro adresářovou službu založenou na serveru LDAP
- `file:/`*`directoryPath`*, pro adresářovou službu založenou na lokálním systému souborů

Adresu 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 zapnuto. Ve výchozím nastavení je trasování vypnuto.

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 v systémech UNIX and Linux a IVTTidy.bat v systému Windowsa je v podadresáři `bin` tříd WebSphere MQ pro instalační adresář 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 “Proměnné prostředí používané třídami produktu IBM WebSphere MQ pro platformu JMS” na stránce 699.
- 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 nakonfigurován pro ukládání objektů Java, nebo nejsou správná oprávnění k objektům nebo přípona. 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 “Příprava a spuštění ukázkových programů” na stránce 104.

Ověřovací test instalace publikování/odběru pro třídy WebSphere MQ pro JMS

Program IVT (publish/subscribe installation verification test) je dodáván s třídami WebSphere MQ pro platformu 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 viz “Typy objektů JMS” na stránce 906.

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 UNIX and Linux a PSIVTRun.bat na systému Windowsa nachází se v podadresáři `bin` tříd WebSphere MQ pro instalační adresář 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 “Příprava a spuštění ukázkových programů” na stránce 104 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 je relevantní pouze v případě, že parametr -v určuje číslo verze správce front menší než 7 a že jako zprostředkovatele publikování/odběru používáte produkt WebSphere Event Broker nebo WebSphere Message Broker.

-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 WebSphere 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 zapnuto. Ve výchozím nastavení je trasování vypnuto.

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
Publish/Subscribe Installation Verification Test
```

```
Creating a TopicConnectionFactory
Creating a Connection
Creating a Session
Creating a Topic
Creating a TopicPublisher
Creating a TopicSubscriber
Creating a TextMessage
Adding text
Publishing the message to topic://MQJMS/PSIVT/Information
Waiting for a message to arrive [5 secs max]...

Got message:
  JMSMessage class: jms_text
  JMSType:          null
  JMSDeliveryMode:  2
  JMSExpiration:    0
  JMSPriority:      4
  JMSMessageID:     ID:414d5120514d5f6d627720202020202001edb14620006706
  JMSTimestamp:     1187182520203
  JMSCorrelationID: ID:414d5120514d5f6d627720202020202001edb14620006704
  JMSDestination:   topic://MQJMS/PSIVT/Information
  JMSReplyTo:       null
  JMSRedelivered:   false
    JMSXUserID: mwhite
    JMS_IBM_Encoding: 273
    JMS_IBM_PutApplType: 26
    JMSXAppID: QM_mbw
    JMSXDeliveryCount: 1
    JMS_IBM_PutDate: 20070815
    JMS_IBM_ConnectionID: 414D5143514D5F6D627720202020202001EDB14620006601
    JMS_IBM_PutTime: 12552020
    JMS_IBM_Format: MQSTR
    JMS_IBM_MsgType: 8
A simple text message from the MQJMSPSIVT program
Reply string equals original string
Closing TopicSubscriber
Closing TopicPublisher
Closing Session
Closing Connection
PSIVT finished
```

Ověřovací test instalace publikování/odběru 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 WebSphere MQ JMS tak, aby mohl používat adresářovou službu k ukládání spravovaných objektů. Další informace o těchto předpokladech viz “Nezbytné předpoklady pro třídy WebSphere MQ pro JMS” na stránce 694. Informace o konfiguraci nástroje pro administraci rozhraní JMS WebSphere MQ naleznete v tématu “Konfigurace nástroje pro administraci služby JMS” na stránce 903.

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 UNIX and Linux a IVTSetup.bat na systému Windowsa nachází se v podadresáři `bin` tříd WebSphere MQ pro instalační adresář JMS. Chcete-li spustit skript, zadejte následující příkaz:

```
IVTSetup
```

Skript vyvolá nástroj administrace WebSphere MQ JMS pro vytvoření spravovaných objektů.

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 spuštěn v režimu klienta, připojte se ke správci front, který není výchozím správcem front, nebo použijte produkt WebSphere Event Broker nebo WebSphere Message Broker namísto vestavěné funkce

publikování/odběru, Chcete-li změnit příslušné vlastnosti objektu továrny MQTopicConnection, musíte použít nástroj administrace JMS WebSphere MQ nebo produkt WebSphere MQ Explorer. Informace o použití nástroje pro administraci produktu WebSphere MQ JMS naleznete v části “Použití nástroje pro administraci produktu WebSphere MQ JMS” na stránce 902. Informace o použití produktu WebSphere MQ Explorer naleznete v nápovědě k produktu WebSphere MQ Explorer.

Objekt MQTopic je svázán s názvem ivtT a je vytvořen s výchozími hodnotami pro všechny jeho vlastnosti s výjimkou vlastnosti TOPIC, která má hodnotu MQJMS/PSIVT/Information.

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*"
Adresa URL (Uniform Resource Locator) adresářové služby. Adresa URL může mít jeden z následujících formátů:

- `ldap://hostname/contextName`, pro adresářovou službu založenou na serveru LDAP
- `file:/directoryPath`, pro adresářovou službu založenou na lokálním systému souborů

Adresu 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 zapnuto. Ve výchozím nastavení je trasování vypnuto.

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 v systémech UNIX and Linux a IVTTidy.bat v systému Windowsa je v podadresáři `bin` tříd WebSphere MQ pro instalační adresář 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 “Proměnné prostředí používané třídami produktu IBM WebSphere MQ pro platformu JMS” na stránce 699.
- 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 nakonfigurován pro ukládání objektů Java, nebo nejsou správná oprávnění k objektům nebo přípona. 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 “Příprava a spuštění ukázkových programů” na stránce 104.

Testovací program ověření instalace pro adaptér prostředků WebSphere MQ

Program IVT je dodáván jako soubor EAR. Chcete-li použít tento program, musíte jej implementovat a definovat některé objekty jako prostředky JCA.

Program pro test verifikace instalace (IVT) je dodáván jako soubor podnikového archivu (EAR) s názvem wmq.jmsra.ivt.ear. Tento soubor je instalován s třídami produktu WebSphere MQ pro službu JMS ve stejném adresáři jako soubor RAR adaptéru prostředků produktu WebSphere MQ , wmq.jmsra.rar. Informace o tom, kde jsou tyto soubory nainstalovány, najdete v tématu “Instalace adaptéru prostředků produktu WebSphere MQ” na stránce 711.

Na svém aplikačním serveru musíte implementovat program IVT. Program IVT obsahuje servlet a objekt MDB, který testuje, zda lze odesílat zprávy do fronty WebSphere MQ a přijímat je od ní. Volitelně můžete použít program IVT k ověření, že adaptér prostředků WebSphere MQ byl správně nakonfigurován pro podporu distribuovaných transakcí.

Než budete moci spustit program IVT, musíte definovat objekt ConnectionFactory , objekt fronty a případně objekt specifikace aktivace jako prostředky JCA a zajistit, aby váš aplikační server vytvořil objekty JMS z těchto definic a svázal je s oborem názvů JNDI. Můžete si vybrat vlastnosti objektů, ale následující sada vlastností je jednoduchý příklad:

Objekt ConnectionFactory

```
channel:              SYSTEM.DEF.SVRCONN
hostName:             localhost
port:                 1414
queueManager:         ExampleQM
transportType:        CLIENT
```

Objekt fronty

```
baseQueueManagerName:  ExampleQM
baseQueueName:         TEST.QUEUE
```

Program IVT standardně očekává, že objekt ConnectionFactory má být svázán v oboru názvů JNDI s názvem jms/ivt/IVTCF a s objektem fronty, který má být svázán s názvem jms/ivt/IVTQueue. Můžete použít různé názvy, ale pokud ano, musíte zadat názvy objektů na počáteční stránce programu IVT a upravit odpovídajícím způsobem soubor EAR.

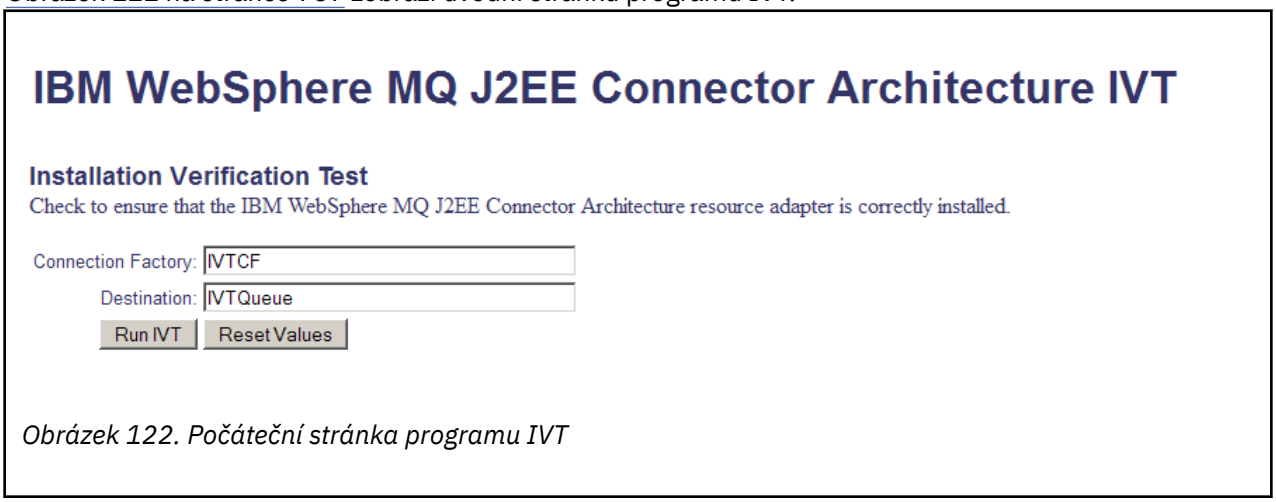
Poté, co jste implementovali program IVT a aplikační server vytvořil objekty platformy JMS a svázal je do oboru názvů JNDI, můžete spustit program IVT zadáním adresy URL do webového prohlížeče v následujícím formátu:

```
http://app_server_host:port/WMQ_IVT/
```

kde *hostitel_aplikačního_serveru* je adresa IP nebo název hostitele systému, na kterém běží váš aplikační server, a *port* je číslo portu TCP, na kterém naslouchá aplikační server. Příklad:

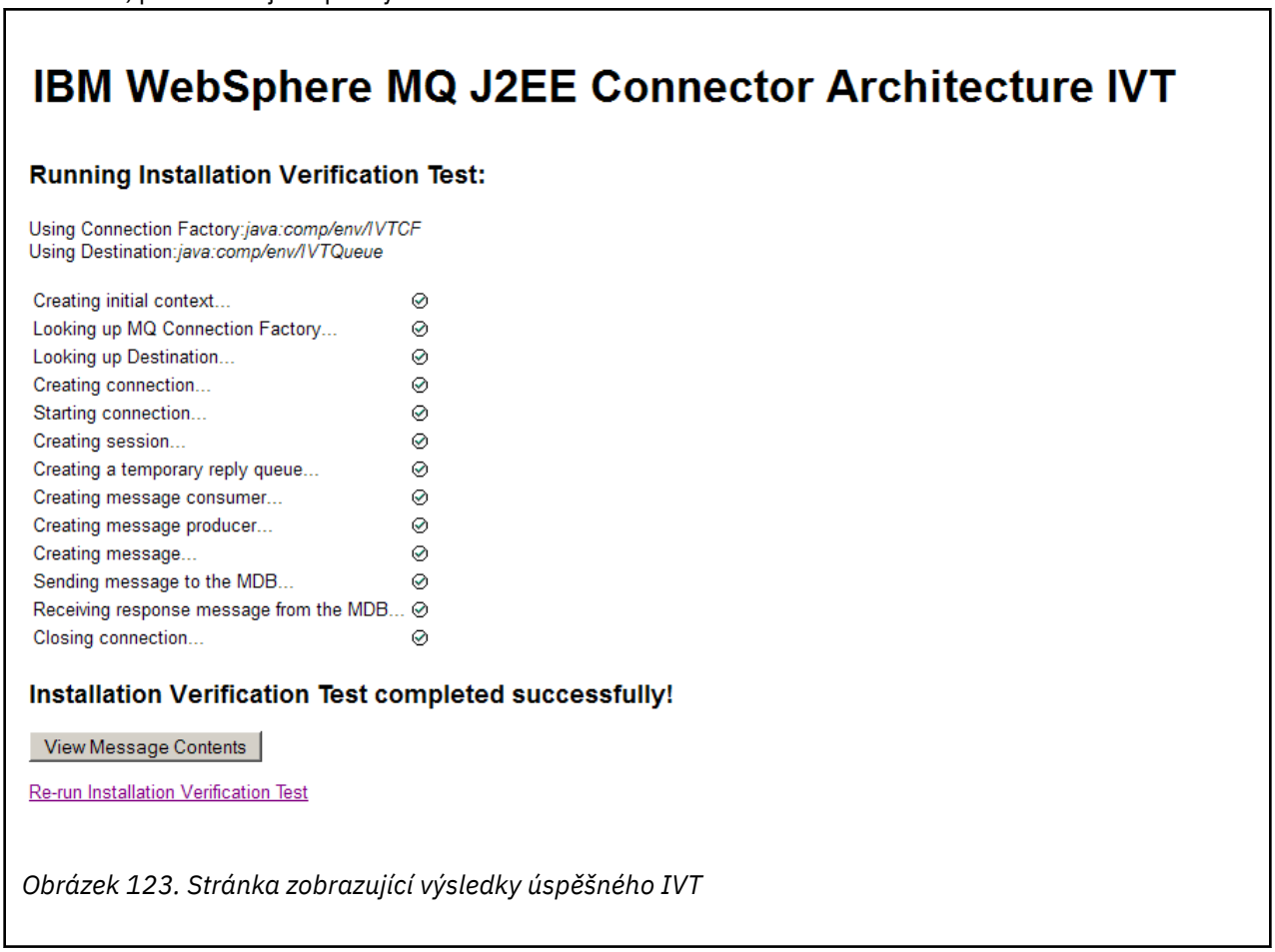
```
http://localhost:9080/WMQ_IVT/
```

Obrázek 122 na stránce 757 zobrazí úvodní stránku programu IVT.



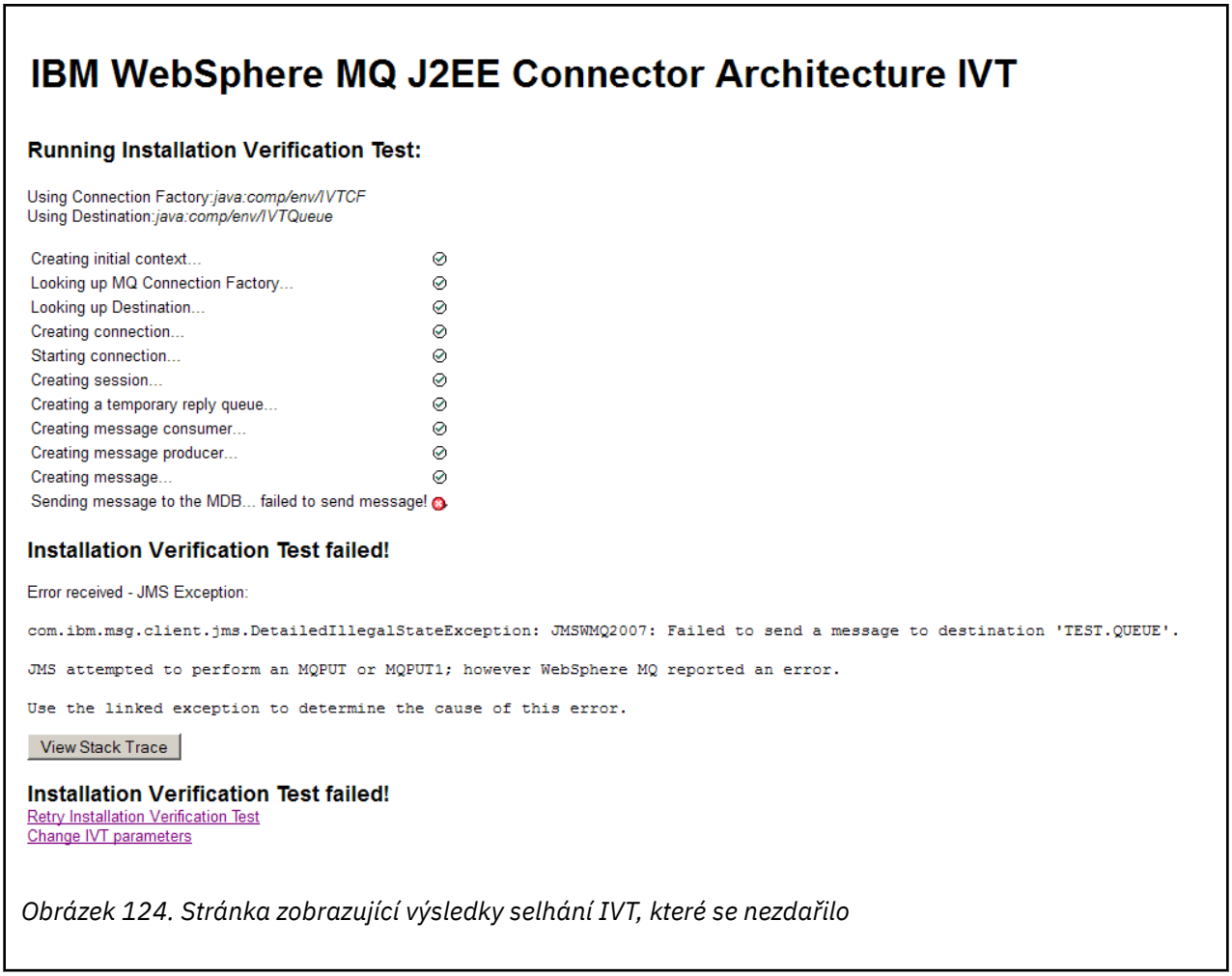
Obrázek 122. Počáteční stránka programu IVT

Chcete-li spustit test, klepněte na volbu **Spustit IVT**. Obrázek 123 na stránce 757 zobrazí stránku, která se zobrazí, pokud IVT je úspěšný.



Obrázek 123. Stránka zobrazující výsledky úspěšného IVT

Pokud se IVT nezdaří, zobrazí se stránka podobná té, která je zobrazena v části Obrázek 124 na stránce 758 . Chcete-li získat další informace o příčině selhání, klepněte na volbu **Zobrazit trasování zásobníku**.



Obrázek 124. Stránka zobrazující výsledky selhání IVT, které se nezdařilo

Podrobné pokyny a informace o obslužných skriptech poskytovaných pro implementaci aplikace IVT na aplikačním serveru JBoss a WAS CE viz:

Související úlohy

“Instalace a testování adaptéru prostředků MQ v produktu WAS CE” na stránce 758
Instalace adaptéru prostředků IBM WebSphere MQ a spuštění aplikace testu verifikace instalace (IVT) v produktu WebSphere Application Server CE.

“Instalace a testování adaptéru prostředků v produktu JBoss AS 5.1 a 6” na stránce 761
Po instalaci adaptéru prostředků produktu IBM WebSphere MQ v systému JBoss AS 5.1 nebo 6 můžete otestovat instalaci adaptéru prostředku nainstalováním a spuštěním aplikace IVT (Installation Verification Test).

Instalace a testování adaptéru prostředků MQ v produktu WAS CE

Instalace adaptéru prostředků IBM WebSphere MQ a spuštění aplikace testu verifikace instalace (IVT) v produktu WebSphere Application Server CE.

Než začnete

Tato úloha předpokládá, že máte spuštěný server WebSphere Application Server CE a že jste obeznámeni se standardními administračními úlohami pro tuto úlohu. Tato úloha také předpokládá, že máte instalaci produktu IBM WebSphere MQ na svém lokálním systému a že jste obeznámeni se standardními úlohami administrace.

Pokud používáte adaptér prostředků pro připojení k klientu produktu IBM WebSphere MQ a potřebujete provést distribuované transakce XA, musíte postupovat podle dalších kroků označených **Pouze klient XA**.

1. Vytvořte správce front s názvem `ExampleQM`a nastavte jej tak, jak je popsáno v tématu “Příprava a spuštění ukázkových programů” na stránce 104 a všimněte si, že modul listener by měl být spuštěn na portu 1414, kanál, který má být použit, se nazývá `SYSTEM.DEF.SVRCONN` a fronta použitá aplikací IVT má název `TEST.QUEUE`. Modelová fronta `SYSTEM.DEFAULT.MODEL.QUEUE` bude také muset být udělena oprávnění DSP a PUT, takže tato aplikace může vytvořit dočasnou frontu odpovědí. Chcete-li použít jiného správce front, jiné podrobnosti o připojení nebo jinou frontu, přečtěte si téma “Implementace aplikace IVT na WAS CE s vlastním prostředím MQ” na stránce 760.
2. Získejte soubor adaptéru prostředků (`wmq.jmsra.rar`), aplikace IVT (`wmq.jmsra.ivt.ear`), a `WAS_CE_jmsra_deployment_plan.xml` a `WAS_CE_jmsra_ivt_deployment_plan.xml` `deployment plan files`. Podrobné informace o umístění těchto souborů najdete v tématu “Instalace adaptéru prostředků produktu WebSphere MQ” na stránce 711.

Popis vazeb a připojení v režimu klienta viz “Režimy připojení pro třídy WebSphere MQ pro JMS” na stránce 747.

Chcete-li použít jinou frontu, správce front, port, uzel, kanál nebo použít režim vazeb namísto režimu klienta, přečtěte si téma “Implementace aplikace IVT na WAS CE s vlastním prostředím MQ” na stránce 760.

Postup

1. **Pouze klient XA**: Upravte svou kopii souboru `WAS_CE_jmsra_deployment_plan.xml` .
   a) Vyhledejte definici připojení jms/ivt/IVTCF a upravte ji tak, aby továrna připojení byla povolena XA-transakce.
      i) Označte jako komentář sekci NonXA :

      ```
      <conn:xa-transaction>
      ```

      ii) Zrušte komentář v sekci konfigurace XA:

      ```
      <conn:xa-transaction>
          <conn:transaction-caching/>
      </conn:xa-transaction>
      ```

   b) Uložte změny.
2. Volitelné: **Pouze pro klienta XA**: Upravte deskriptor sestavení objektu MDB tak, aby vyžadoval transakce. Vynutí objekt MDB v IVT, aby se účastnil transakce XA, ačkoli aplikace IVT stále pracuje bez této úpravy.
   a) Otevřete soubor `wmq.jmsra.ivt.ear` .
   b) Otevřete v něm soubor `WMQ_IVT_MDB.jar` .
   c) Upravte `META-INF/ejb-jar.xml`.
      i) Označte jako komentář nebo odstraňte řádek v rámci deskriptoru sestavení:

      ```
      <trans-attribute>NotSupported</trans-attribute>
      ```

      ii) Odstraňte znak komentáře řádku v rámci deskriptoru sestavení:

      ```
      <trans-attribute>Required</trans-attribute>
      ```

      iii) Uložte provedené změny a aktualizujte soubor v souboru `WMQ_IVT_MDB.jar` .
      iv) Aktualizujte soubor wmq.jmsra.ivt.ear pomocí upraveného souboru `WMQ_IVT_MDB.jar` .
3. Implementujte adaptér prostředků na váš server s použitím upraveného souboru s plánem implementace.
   a) Chcete-li to provést na příkazovém řádku, zadejte následující příkaz WAS CE:

   ```
   deploy -u system -p manager deploy wmq.jmsra.rar WAS_CE_jmsra_deployment_plan.xml
   ```

b) Pomocí rozhraní webové administrace přejděte na volbu **Aplikace > Implementátor:**

i) Nastavte archiv tak, aby byl souborem wmq.jmsra.rar .

ii) Nastavte plán na soubor WAS_CE_jmsra_deployment_plan.xml .

iii) Ujistěte se, že je vybrána volba 'Spustit aplikaci po instalaci'.

iv) Klepněte na volbu **Instalovat**.

4. Implementujte aplikaci IVT na váš server pomocí poskytnutého plánu nasazení.

a) Na příkazovém řádku lze toto provést pomocí následujícího příkazu WAS CE:

```
deploy -u system -p manager deploy wmq.jmsra.ivt.ear WAS_CE_jmsra_ivt_deployment_plan.xml
```

b) Pomocí rozhraní webové administrace přejděte na volbu **Aplikace > Implementátor:**

i) Nastavte volbu **Archivovat** tak, aby se souborem `wmq.jmsra.ivt.ear` .

ii) Nastavte **Plán** tak, aby byl souborem `WAS_CE_jmsra_ivt_deployment_plan.xml` .

iii) Ujistěte se, že je vybrána volba **Spustit aplikaci po instalaci** .

iv) Klepněte na volbu **Instalovat**.

5. Spusťte aplikaci IVT. Další informace naleznete v tématu “Testovací program ověření instalace pro adaptér prostředků WebSphere MQ” na stránce 756. Pro WAS CE je výchozí adresa URL `http://localhost:8080/WMQ_IVT/`.

Implementace aplikace IVT na WAS CE s vlastním prostředím MQ

Chcete-li použít jinou frontu, správce front, port, hostitele, kanál nebo použít režim vazeb namísto režimu klienta, je třeba před implementací adaptéru prostředků nebo aplikace IVT upravit aplikaci IVT a přidružené skripty v produktu WebSphere Application Server CE.

Informace o této úloze

Chcete-li implementovat do jiné konfigurace než v produktu “Instalace a testování adaptéru prostředků MQ v produktu WAS CE” na stránce 758, tj. chcete-li použít jinou frontu, správce front, port, hostitele, kanál nebo použít režim vazeb namísto režimu klienta, proveďte před implementací adaptéru prostředků nebo aplikace IVT následující kroky.

Postup

1. Chcete-li určit jiného správce front a frontu, která má být použita pro aplikaci IVT, nastavte hodnoty pro správce front a frontu v produktu `WAS_CE_jmsra_deployment_plan.xml`, podrobnosti viz “Nastavení hodnot pro správce front a frontu” na stránce 761.
2. Chcete-li určit jiného správce front a frontu v konfiguraci pro objekt MDB (message-driven bean), nastavte hodnoty pro správce front a frontu, kterou používáte v produktu `WAS_CE_jmsra_ivt_deployment_plan.xml`, podrobné informace viz “Nastavení hodnot pro konfiguraci objektu MDB” na stránce 761.
3. Pokud konfigurujete adaptér prostředků pro připojení k produktu IBM WebSphere MQ v režimu vazeb, ujistěte se, že knihovny JNI jsou na systémové cestě nebo v cestě pro WAS CE. Další informace viz “Instalace a testování adaptéru prostředků MQ v produktu WAS CE” na stránce 758.
4. Pokud jste adaptér prostředků již naimplementovali, můžete jej znovu implementovat s upraveným plánem implementace, chcete-li změnit nastavení pomocí následujícího příkazu:

```
deploy --user system --password manager redeploy wmq.jmsra.rar
WAS_CE_jmsra_deployment_plan.xml
```

Jak pokračovat dále

Pokračujte v implementaci adaptéru prostředků, jak je popsáno v tématu “Instalace a testování adaptéru prostředků MQ v produktu WAS CE” na stránce 758.

Nastavení hodnot pro správce front a frontu

Vysvětluje, jak nastavit hodnoty pro správce front a frontu, které používáte v produktu `WAS_CE_jmsra_deployment_plan.xml`.

Postup

V produktu `WAS_CE_jmsra_deployment_plan.xml`nastavte hodnoty pro správce front a frontu, které používáte pro aplikaci IVT.

Pro definici připojení jms/ivt/IVTCF:

1. Nastavte hodnotu prvku queueManager tak, aby se jmenoval názvem správce front.
2. Používáte-li připojení klienta, nastavte hodnotu různých prvků připojení klienta tak, aby byla vhodná pro připojení ke správci front.
3. Používáte-li připojení vazeb:
   a. Nastavte hodnotu prvku transportType na hodnotu BINDINGS.
   b. Označte jako komentář nebo odstraňte různé prvky připojení klienta.
4. Pro místo určení zpráv jms/ivt/IVTQueue nastavte hodnotu prvku Název baseQueuena název fronty, kterou jste vytvořili pro aplikaci IVT
5. Uložte změny.

Nastavení hodnot pro konfiguraci objektu MDB

Vysvětluje, jak nastavit hodnoty pro konfiguraci objektu MDB v produktu `WAS_CE_jmsra_deployment_plan.xml`.

Postup

V produktu `WAS_CE_jmsra_ivt_deployment_plan.xml` nastavte hodnoty pro správce front a frontu, které používáte v konfiguraci pro objekt MDB.

Pro objekt typu message-driven bean WMQ_IVT_MDB:

1. Nastavte hodnotu prvku queueManager tak, aby se jmenoval názvem správce front.
2. Používáte-li připojení klienta, nastavte hodnotu různých prvků připojení klienta tak, aby byla vhodná pro připojení ke správci front.
3. Používáte-li připojení vazeb:
   a. Nastavte hodnotu prvku transportType na hodnotu BINDINGS.
   b. Označte jako komentář nebo odstraňte různé prvky připojení klienta.
4. Uložte změny.

Instalace a testování adaptéru prostředků v produktu JBoss AS 5.1 a 6

Po instalaci adaptéru prostředků produktu IBM WebSphere MQ v systému JBoss AS 5.1 nebo 6 můžete otestovat instalaci adaptéru prostředku nainstalováním a spuštěním aplikace IVT (Installation Verification Test).

Než začnete

Důležité: Tyto pokyny jsou určeny pro JBoss AS 5.1 a 6, nejsou platné pro JBoss AS 7.

Informace o instalaci adaptéru prostředků v produktu JBoss EAP 6.3viz “Instalace a testování adaptéru prostředků v produktu JBoss EAP 6.3” na stránce 764.

Tato úloha předpokládá, že máte spuštěný server JBoss a jsou obeznámeni se standardními úlohami administrace pro tento server. Tato úloha také předpokládá, že máte instalaci produktu IBM WebSphere MQ na svém lokálním systému a že jste obeznámeni se standardními úlohami administrace.

Pokud používáte adaptér prostředků pro připojení k klientu produktu IBM WebSphere MQ a potřebujete provést distribuované transakce XA, musíte postupovat podle dalších kroků označených **Pouze klient XA**. Popis vazeb a připojení v režimu klienta viz "Režimy připojení pro třídy WebSphere MQ pro JMS" na stránce 747.

Postup

1. Vytvořte správce front s názvem ExampleQMa nastavte jej tak, jak je popsáno v tématu "Příprava a spuštění ukázkových programů" na stránce 104.

   Při nastavování správce front vezměte v úvahu následující body:

   - Modul listener musí být spuštěn na portu 1414.
   - Kanál, který má být použit, se nazývá SYSTEM.DEF.SVRCONN.
   - Fronta používaná aplikací IVT má název TEST.QUEUE.

   Modelová fronta SYSTEM.DEFAULT.MODEL.QUEUE je také třeba udělit oprávnění DSP a PUT, takže tato aplikace může vytvořit dočasnou frontu odpovědí.

   Chcete-li použít jiného správce front, jiné podrobnosti o připojení nebo jinou frontu, přečtěte si téma "Implementace aplikace IVT na WAS CE s vlastním prostředím MQ" na stránce 760.
2. Získejte informace o souboru adaptéru prostředků (`wmq.jmsra.rar`), aplikaci IVT (`wmq.jmsra.ivt.ear`) a souboru `jboss-jmsra-ds.xml` .

   Informace o umístění těchto souborů najdete v tématu "Instalace adaptéru prostředků produktu WebSphere MQ" na stránce 711.
3. **Pouze pro klienta XA**: Upravte soubor `jboss-jmsra-ds.xml` tak, aby povolovali transakce XA na faktorii připojení.
   a) Označte jako komentář nebo odstraňte řádek v definici továrny připojení `<local-transaction/>`.
   b) Okomentujte řádek v rámci definice továrny připojení `<xa-transaction/>`.
   c) Uložte změny.
4. **Pouze pro klienta XA**: (volitelné) Upravte deskriptor sestavení objektu MDB tak, aby vyžadoval transakce. Vynutí objekt MDB v IVT, aby se účastnil transakce XA, ačkoli aplikace IVT stále pracuje bez této úpravy.
   a) Otevřete soubor `wmq.jmsra.ivt.ear` .
   b) Otevřete v něm soubor `WMQ_IVT_MDB.jar` .
   c) Upravte `META-INF/ejb-jar.xml`:
      i) Označte jako komentář nebo odstraňte řádek v rámci deskriptoru sestavení:

      ```
      <trans-attribute>NotSupported</trans-attribute>
      ```

      ii) Odstraňte znak komentáře řádku v rámci deskriptoru sestavení:

      ```
      <trans-attribute>Required</trans-attribute>
      ```

      iii) Uložte provedené změny a aktualizujte soubor v souboru `WMQ_IVT_MDB.jar` .
      iv) Aktualizujte soubor `wmq.jmsra.ivt.ear` s upraveným `WMQ_IVT_MDB.jar`.
5. Naimplementujte adaptér prostředků na váš server tak, že zkopírujete soubor `wmq.jmsra.rar` do adresáře `jboss/server/default/deploy`.
6. Vytvořte prostředky JMS požadované pro aplikaci IVT zkopírováním souboru `jboss-jmsra-ds.xml` do adresáře `jboss/server/default/deploy`.
7. Implementujte aplikaci IVT okopírováním souboru `wmq.jmsra.ivt.ear` do adresáře `jboss/server/default/deploy`.

8. Spusťte aplikaci IVT. Další informace naleznete v tématu “Testovací program ověření instalace pro adaptér prostředků WebSphere MQ” na stránce 756. Pro produkt JBossje výchozí adresa URL `http://localhost:8080/WMQ_IVT/`.

Implementace aplikace IVT v produktu JBoss s vlastním prostředím produktu IBM WebSphere MQ

Při instalaci adaptéru prostředků produktu IBM WebSphere MQ v produktu JBoss, pokud chcete použít jinou frontu, správce front, port, hostitele, kanál nebo použít režim vazeb namísto režimu klienta, je třeba nejprve upravit aplikaci IVT a přidružené skripty v produktu JBoss před implementací adaptéru prostředků nebo aplikace IVT.

Informace o této úloze

Důležité: Tyto pokyny jsou použitelné pouze pro prostředí Java EE verze 6 a 5, nikoli pro prostředí Java EE verze 7. Použití těchto pokynů pro produkt JBoss verze 8 (WildFly) proto není podporováno.

Chcete-li implementovat do jiné konfigurace než zadané v produktu “Instalace a testování adaptéru prostředků v produktu JBoss AS 5.1 a 6” na stránce 761, tj. chcete-li namísto režimu klienta používat jiného správce front, frontu, port, hostitele, kanál nebo použít režim vazeb, pak před implementací adaptéru prostředků nebo aplikace IVT proveďte následující kroky.

Postup

1. Chcete-li určit jiného správce front a frontu, která má být použita pro aplikaci IVT, nastavte hodnoty pro správce front a frontu.
   a) Pro definici připojení jms/ivt/IVTCF:
      i) Nastavte hodnotu vlastnosti konfigurace queueManager tak, aby se jmenoval názvem správce front.
      ii) Používáte-li připojení klienta, nastavte hodnotu různých prvků připojení klienta tak, aby byla vhodná pro připojení ke správci front.
      iii) Používáte-li připojení vazeb, nastavte hodnotu prvku transportType na hodnotu BINDINGS a poté označte jako komentář nebo odstraňte různé prvky připojení klienta.
   b) Pro objekty bean jms/ivnt/IVTQueue nastavte hodnotu prvku Název baseQueuena název fronty, kterou jste vytvořili pro aplikaci IVT.
   c) Uložte změny.
2. Chcete-li určit jiného správce front a frontu v konfiguraci pro objekt MDB (message-driven bean), upravte konfiguraci objektu MDB tak, aby se připojoval ke správci front a frontě.
   a) Otevřete soubor `wmq.jmsra.ivt.ear` .
   b) Otevřete `WMQ_IVT_MDB.jar` v něm.
   c) Upravte `META-INF/ejb-jar.xml`:
      i) Nastavte hodnotu vlastnosti activation-config správce front queueManager na název správce front.
      ii) Používáte-li připojení klienta, nastavte hodnotu různých vlastností aktivace připojení klienta tak, aby byla vhodná pro připojení ke správci front.
      iii) Používáte-li připojení vazeb, nastavte hodnotu vlastnosti activation-config transportType na hodnotu BINDINGS a poté označte jako komentář nebo odstraňte různé prvky připojení klienta.
   d) Uložte změny a aktualizujte soubor v souboru `WMQ_IVT_MDB.jar` .
   e) Aktualizujte soubor `wmq.jmsra.ivt.ear` s upraveným `WMQ_IVT_MDB.jar`.
3. Pokud konfigurujete adaptér prostředků pro připojení k produktu IBM WebSphere MQ v režimu vazeb, ujistěte se, že knihovny JNI jsou na systémové cestě, nebo cestu pro JBoss. Podrobné informace naleznete v tématu “Konfigurace knihoven JNI (Java Native Interface)” na stránce 701.

Jak pokračovat dále

Pokračujte v implementaci adaptéru prostředků, jak je popsáno v tématu “Instalace a testování adaptéru prostředků v produktu JBoss AS 5.1 a 6” na stránce 761.

.

Instalace a testování adaptéru prostředků v produktu JBoss EAP 6.3

Po instalaci adaptéru prostředků produktu IBM WebSphere MQ v produktu JBoss Enterprise Application Platform (EAP) 6.3na samostatném serveru nebo na serveru spuštěném ve spravované doméně můžete otestovat instalaci adaptéru prostředku instalováním a spuštěním aplikace IVT (Installation Verification Test).

Informace o této úloze

Důležité: Tyto pokyny jsou určeny pouze pro produkt JBoss EAP 6.3 . Informace o instalaci adaptéru prostředků v příručce JBoss AS 5.1 a 6 naleznete v příručce “Instalace a testování adaptéru prostředků v produktu JBoss AS 5.1 a 6” na stránce 761.

Tato úloha předpokládá, že máte spuštěný server JBoss a jsou obeznámeni se standardními úlohami administrace pro tento server. Tato úloha také předpokládá, že máte instalaci produktu IBM WebSphere MQ na svém lokálním systému a že jste obeznámeni se standardní administrací.

Postup

1. Vytvořte správce front s názvem ExampleQMa nastavte jej tak, jak je popsáno v tématu “Příprava a spuštění ukázkových programů” na stránce 104.

   Při nastavování správce front vezměte v úvahu následující body:

   - Modul listener musí být spuštěn na portu 1414.
   - Kanál, který má být použit, se nazývá SYSTEM.DEF.SVRCONN.
   - Fronta používaná aplikací IVT má název TEST.QUEUE.

   Modelová fronta SYSTEM.DEFAULT.MODEL.QUEUE je také třeba udělit oprávnění DSP a PUT, takže tato aplikace může vytvořit dočasnou frontu odpovědí.

   Chcete-li použít jiného správce front, jiné podrobnosti o připojení nebo jinou frontu, přečtěte si téma “Implementace aplikace IVT na WAS CE s vlastním prostředím MQ” na stránce 760.
2. Získejte informace o souboru adaptéru prostředků (`wmq.jmsra.rar`) a aplikaci IVT (`wmq.jmsra.ivt.ear`).

   Informace o umístění těchto souborů najdete v tématu “Instalace adaptéru prostředků produktu WebSphere MQ” na stránce 711.
3. Instalujte adaptér prostředků a potom otestujte instalaci spuštěním aplikace pro verifikaci instalace (IVT):

   - Pokud instalujete adaptér prostředků na samostatný server, prohlédněte si téma “Instalace a testování na samostatném serveru” na stránce 764.
   - Instalujete-li adaptér prostředků na server spuštěný ve spravované doméně, přečtěte si téma “Instalace a testování na serveru spuštěném ve spravované doméně” na stránce 766.

Instalace a testování na samostatném serveru

Po instalaci adaptéru prostředků produktu IBM WebSphere MQ v produktu JBoss EAP 6.3 na samostatném serveru můžete otestovat instalaci adaptéru prostředku instalováním a spuštěním aplikace IVT (Installation Verification Test).

Informace o této úloze

Informace v této úloze slouží k instalaci a testování adaptéru prostředků na samostatném serveru. Instalujete-li adaptér prostředků na server spuštěný ve spravované doméně, přečtěte si téma “Instalace a testování na serveru spuštěném ve spravované doméně” na stránce 766.

Důležité: Tyto pokyny jsou určeny pouze pro produkt JBoss EAP 6.3 . Informace o instalaci adaptéru prostředků v příručce JBoss AS 5.1 a 6 naleznete v příručce “Instalace a testování adaptéru prostředků v produktu JBoss AS 5.1 a 6” na stránce 761.

Postup

1. Naimplementujte adaptér prostředků na váš server tak, že zkopírujete soubor `wmq.jmsra.rar` do adresáře <*EAP_HOME*>`/standalone/deployments`.
2. Vytvořte prostředky JMS požadované pro aplikaci IVT přidáním následujících položek do sekce `<resource-adapters>` souboru <*EAP_HOME*>`/standalone/configuration/standalone-full.xml` :

```
<resource-adapter id="wmq.jmsra">
  <archive>
      wmq.jmsra.rar
  </archive>
  <transaction-support>NoTransaction</transaction-support>
  <connection-definitions>
     <connection-definition
          class-name="com.ibm.mq.connector.outbound.ManagedConnectionFactoryImpl"
          jndi-name="java:jboss/jms/ivt/IVTCF"
          enabled="true"
          use-java-context="true"
          pool-name="IVTCF">
       <config-property name="port">
           1414
       </config-property>
       <config-property name="hostName">
           localhost
       </config-property>
       <config-property name="channel">
           SYSTEM.DEF.SVRCONN
       </config-property>
       <config-property name="transportType">
           CLIENT
       </config-property>
       <config-property name="queueManager">
           ExampleQM
       </config-property>
     </connection-definition>
  </connection-definitions>
  <admin-objects>
    <admin-object class-name="com.ibm.mq.connector.outbound.MQQueueProxy"
                  jndi-name="java:jboss/jms/ivt/IVTQueue"
                  pool-name="IVTQueue">
      <config-property name="baseQueueName">
          TEST.QUEUE
      </config-property>
    </admin-object>
  </admin-objects>
</resource-adapter>
```

3. Přidejte následující informace do parametrů spuštění aplikačního serveru:

```
-Dcom.ibm.mq.connector.IVTMDBCFJNDIName=java:jboss/jms/ivt/IVTCF
```

4. Implementujte aplikaci IVT zkopírováním souboru `wmq.jmsra.ivt.ear` do adresáře <*EAP_HOME*>`/standalone/deployments`.
5. Spusťte aplikační server.
6. Spusťte aplikaci IVT.

   Další informace viz “Testovací program ověření instalace pro adaptér prostředků WebSphere MQ” na stránce 756. Pro produkt JBossje výchozí adresa URL `http://localhost:8080/WMQ_IVT/`.

Poznámka: Názvy JNDI použité pro prostředky JMS požadované pro spuštění aplikace IVT jsou následující:

```
Connection Factory : IVTCF
JNDI name          : java:jboss/jms/ivt/IVTCF

Destination        : IVTQueue
JNDI name          : java:jboss/jms/ivt/IVTQueue
```

Spustíte-li aplikaci IVT pomocí výše uvedené adresy URL, zadejte názvy JNDI těchto prostředků do příslušných polí a poté spusťte aplikaci klepnutím na tlačítko **Spustit IVT**.

Instalace a testování na serveru spuštěném ve spravované doméně

Po instalaci adaptéru prostředků produktu IBM WebSphere MQ v produktu JBoss EAP 6.3 na serveru spuštěném ve spravované doméně můžete otestovat instalaci adaptéru prostředku instalováním a spuštěním aplikace IVT (Installation Verification Test).

Informace o této úloze

Informace v této úloze slouží k instalaci a testování adaptéru prostředků na serveru spuštěném ve spravované doméně. Pokud instalujete adaptér prostředků na samostatný server, prohlédněte si téma “Instalace a testování na samostatném serveru” na stránce 764.

Důležité: Tyto pokyny jsou určeny pouze pro produkt JBoss EAP 6.3 . Informace o instalaci adaptéru prostředků v příručce JBoss AS 5.1 a 6 naleznete v příručce “Instalace a testování adaptéru prostředků v produktu JBoss AS 5.1 a 6” na stránce 761.

Postup

1. Implementujte adaptér prostředků na váš server pomocí konzoly správy JBoss nebo rozhraní CLI správy.
2. Vytvořte prostředky JMS požadované pro aplikaci IVT přidáním následujících položek do sekce `<resource-adapters>` v `<EAP_HOME>/domain/configuration/domain.xml file`:

```
<resource-adapter id="wmq.jmsra">
  <archive>
      wmq.jmsra.rar
  </archive>
  <transaction-support>NoTransaction</transaction-support>
  <connection-definitions>
    <connection-definition
         class-name="com.ibm.mq.connector.outbound.ManagedConnectionFactoryImpl"
         jndi-name="java:jboss/jms/ivt/IVTCF"
         enabled="true"
         use-java-context="true"
         pool-name="IVTCF">
      <config-property name="port">
          1414
      </config-property>
      <config-property name="hostName">
          localhost
      </config-property>
      <config-property name="channel">
          SYSTEM.DEF.SVRCONN
      </config-property>
      <config-property name="transportType">
          CLIENT
      </config-property>
      <config-property name="queueManager">
          ExampleQM
      </config-property>
    </connection-definition>
  </connection-definitions>
  <admin-objects>
    <admin-object class-name="com.ibm.mq.connector.outbound.MQQueueProxy"
                  jndi-name="java:jboss/jms/ivt/IVTQueue"
                  pool-name="IVTQueue">
      <config-property name="baseQueueName">
          TEST.QUEUE
      </config-property>
```

```
    </admin-object>
  </admin-objects>
</resource-adapter>
```

3. Přidejte následující informace do parametrů spuštění aplikačního serveru:

```
-Dcom.ibm.mq.connector.IVTMDBCFJNDIName=java:jboss/jms/ivt/IVTCF
```

4. Zastavte a restartujte aplikační server.
5. Implementujte aplikaci IVT pomocí konzoly správy správy JBoss nebo rozhraní CLI správy.
6. Spusťte aplikaci IVT.

   Další informace viz “Testovací program ověření instalace pro adaptér prostředků WebSphere MQ” na stránce 756. Pro produkt JBossje výchozí adresa URL `http://localhost:8080/WMQ_IVT/`.

   Poznámka: Názvy JNDI použité pro prostředky JMS požadované pro spuštění aplikace IVT jsou následující:

```
Connection Factory : IVTCF
JNDI name          : java:jboss/jms/ivt/IVTCF

Destination        : IVTQueue
JNDI name          : java:jboss/jms/ivt/IVTQueue
```

   Zadejte názvy rozhraní JNDI těchto prostředků ve svých příslušných polích při spuštění aplikace IVT pomocí výše uvedené adresy URL a poté spusťte aplikaci klepnutím na tlačítko **Spustit IVT**.

Skripty poskytnuté s třídami produktu WebSphere MQ pro službu JMS

Je k dispozici řada skriptů, které pomáhají s běžnými úlohami, které je třeba provést při použití tříd produktu WebSphere MQ pro službu JMS.

Tabulka 102 na stránce 767 vypíše všechny skripty a jejich použití. Skripty jsou umístěny v podadresáři `bin` instalačního adresáře WebSphere MQ pro instalační adresář platformy JMS.

Tabulka 102. Skripty poskytnuté s třídami produktu WebSphere MQ pro službu JMS

Utility	**Použití**
Vyčištění[1]	Tento skript je udržován kvůli kompatibilitě s předchozími vydáními, ale neprovede žádnou funkci. Ruční vyčištění informací o odběru již není nezbytné
DefaultConfiguration	Spustí výchozí konfigurační aplikaci na platformách jiných než Windows.
formatLog[1] .	Tento skript je udržován kvůli kompatibilitě s předchozími vydáními, ale neprovede žádnou funkci. Výstup protokolu se nyní vytváří v čitelném textu.
IVTRun[1] IVTSetup[1] IVTTidy[1]	Používá se v testu ověření po pevné lince, jak je popsáno v tématu “Ověřovací test dvoubodové instalace pro třídy WebSphere MQ pro JMS” na stránce 749.
JMSAdmin[1]	Spouští nástroj pro administraci produktu WebSphere MQ JMS, jak je popsáno v tématu “Vyvolání nástroje pro administraci produktu IBM WebSphere MQ classes for JMS” na stránce 902.
JMSAdmin.config	Konfigurační soubor pro administrativní nástroj produktu WebSphere MQ JMS, jak je popsáno v tématu “Konfigurace nástroje pro administraci služby JMS” na stránce 903.
PIVTRun[1]	Spustí testovací program pro ověření instalace publikování/odběru, jak je popsáno v tématu “Ověřovací test instalace publikování/odběru pro třídy WebSphere MQ pro JMS” na stránce 752.

Tabulka 102. Skripty poskytnuté s třídami produktu WebSphere MQ pro službu JMS (pokračování)	
Utility	**Použití**
PSReportDump.class	Tato třída je udržována kvůli kompatibilitě s předchozími vydáními, ale neprovádí žádnou funkci.
setjmsenv	Nastaví proměnné prostředí pro spuštění tříd produktu WebSphere MQ pro aplikaci JMS v 32bitovém prostředí JVM (Java virtual machine) v systémech UNIX and Linux , jak je popsáno v tématu “Proměnné prostředí používané třídami produktu IBM WebSphere MQ pro platformu JMS” na stránce 699.
setjmsenv64	Nastaví proměnné prostředí pro spuštění tříd produktu WebSphere MQ pro aplikaci JMS v 64bitovém prostředí JVM na systémech UNIX and Linux , jak je popsáno v tématu “Proměnné prostředí používané třídami produktu IBM WebSphere MQ pro platformu JMS” na stránce 699.
Poznámka: 1. V systému Windowsmá název souboru příponu .bat.	

Podpora pro OSGi

OSGi poskytuje rámec, který podporuje implementaci aplikací jako balíky. Devět svazků balíků OSGi je dodáváno jako součást produktu IBM WebSphere MQ classes for JMS .

OSGi poskytuje obecný účel, zabezpečený a spravovaný rámecJava , který podporuje implementaci aplikací, které jsou dodávány ve formě balíků. Zařízení vyhovující specifikaci OSGi mohou stahovat a instalovat balíky a odebírat je, pokud již nejsou zapotřebí. Rámec spravuje instalaci a aktualizaci balíků v dynamickém a přizpůsobitelném stylu.

IBM WebSphere MQ classes for JMS. zahrnuje následující svazky balíků OSGi.

com.ibm.msg.client.osgi.jms< číslo verze > .jar
Společná vrstva kódu v produktu IBM WebSphere MQ classes for JMS. Informace o vrstvené architektuře tříd produktu WebSphere MQ pro službu JMS naleznete v tématu “Vrstvená architektura” na stránce 772.

com.ibm.msg.client.osgi.jms.prereq_< číslo verze > .jar
Předpokládané soubory JAR (archiv Java) pro obecnou vrstvu.

com.ibm.msg.client.osgi.commonservices.j2se_<version číslo > .jar
Obecné služby pro aplikace Java Platform, Standard Edition (Java SE).

com.ibm.msg.client.osgi.nls_< číslo verze > .jar
Zprávy pro obecnou vrstvu.

com.ibm.msg.client.osgi.wmq_< číslo verze > .jar
Poskytovatel systému zpráv produktu IBM WebSphere MQ v produktu IBM WebSphere MQ classes for JMS. Informace o vrstvé architektuře produktu IBM WebSphere MQ classes for JMS viz “Vrstvená architektura” na stránce 772.

com.ibm.msg.client.osgi.wmq.prereq_< číslo verze > .jar
Předpokládané soubory JAR pro poskytovatele systému zpráv produktu IBM WebSphere MQ .

com.ibm.msg.client.osgi.wmq.nls_< číslo verze > .jar
Zprávy pro poskytovatele systému zpráv produktu IBM WebSphere MQ .

com.ibm.mq.osgi.directip_< číslo_verze > .jar
Soubory JAR, které umožňují poskytovateli systému zpráv produktu IBM WebSphere MQ vytvořit v reálném čase připojení k danému zprostředkovateli.

kde < číslo verze > je číslo verze produktu WebSphere MQ , který byl nainstalován.

Balíky jsou instalovány do podadresáře `java/lib/OSGi` instalace produktu WebSphere MQ nebo složky `java\lib\OSGi` v systému Windows.

Balík com.ibm.mq.osgi.java < číslo verze > .jar, který je nainstalován také do podadresáře `java/lib/OSGi` instalace produktu WebSphere MQ nebo složka produktu `java\lib\OSGi` v systému Windows, je součástí tříd WebSphere MQ pro jazyk Java. Tento balík se nesmí načíst do běhového prostředí OSGi, které má načtené třídy WebSphere MQ pro službu JMS.

Balíky OSGi pro třídy WebSphere MQ pro JMS byly zapsány do specifikace OSGi Release 4. Nepracují v prostředí OSGi Release 3.

Musíte správně nastavit systémovou cestu nebo cestu ke knihovně tak, aby běhové prostředí OSGi mohlo najít všechny požadované soubory DLL nebo sdílené knihovny.

Použijete-li balíky OSGi pro produkt IBM WebSphere MQ classes for JMS, dočasná témata nebudou fungovat. Kromě toho nejsou třídy uživatelské procedury kanálu zapsané v produktu Java podporovány kvůli inherentním problémům při načítání tříd v prostředí zavaděče více tříd, jako je např. OSGi. Balík uživatelů může být informován o třídách IBM WebSphere MQ pro balíky platformy JMS, ale balíky IBM WebSphere MQ classes for JMS si nejsou vědomy žádného balíku uživatele. V důsledku toho zavaděč tříd použitý v balíku IBM WebSphere MQ classes for JMS nemůže načíst třídu ukončení kanálu, která je v uživatelském balíku.

Další informace o specifikaci OSGi naleznete na webu OSGi Alliance .

Řešení problémů s třídami produktu IBM WebSphere MQ pro platformu JMS

Můžete vyšetřit problémy spuštěním programů pro ověření instalace a pomocí zařízení pro trasování a protokolování.

Pokud program nebyl úspěšně dokončen, spusťte jeden z programů pro ověření instalace, jak je popsáno v “Ověřovací test dvoubodové instalace pro třídy WebSphere MQ pro JMS” na stránce 749 a “Ověřovací test instalace publikování/odběru pro třídy WebSphere MQ pro JMS” na stránce 752, a postupujte podle doporučení uvedených v diagnostických zprávách.

Protokolování a IBM WebSphere MQ classes for JMS

Při výchozím nastavení je výstup protokolu odeslán do souboru `mqjms.log` . Můžete ji přesměrovat do specifického souboru nebo adresáře.

Poskytovaná služba pro protokolování produktu IBM WebSphere MQ classes for JMS je poskytována za účelem ohlášení vážných problémů, zejména problémů, které mohou označovat chyby konfigurace spíše než chyby v programování. Výstup protokolu je standardně odeslán do souboru `mqjms.log` v pracovním adresáři prostředí JVM.

Chcete-li přesměrovat výstup protokolu do jiného souboru, nastavte vlastnost com.ibm.msg.client.commonservices.log.outputName. Hodnota této vlastnosti může být:

- Název jedné cesty.
- Čárkami oddělený seznam úplných názvů (všechna data jsou protokolována do všech souborů).

Každý název cesty může být:

- Absolutní nebo relativní.
- `stderr` nebo `System.err` představují standardní chybový proud.
- `sttdout` nebo `System.out` představují standardní výstupní proud.

Pokud hodnota vlastnosti identifikuje adresář, výstup protokolu se zapisuje do `mqjms.log` v tomto adresáři. Pokud hodnota vlastnosti identifikuje určitý soubor, výstup protokolu se zapíše do tohoto souboru.

Tuto vlastnost můžete nastavit v konfiguračním souboru IBM WebSphere MQ classes for JMS nebo jako systémovou vlastnost u příkazu **java** . V následujícím příkladu je vlastnost nastavena jako systémová vlastnost a identifikuje specifický soubor:

```
java -Djava.library.path=library_path
     -Dcom.ibm.msg.client.commonservices.log.outputName=/mydir/mylog.txt
      MyAppClass
```

V příkazu *library_path* je cesta k adresáři obsahujícímu knihovny IBM WebSphere MQ classes for JMS (viz “Konfigurace knihoven JNI (Java Native Interface)” na stránce 701).

Výstup protokolu můžete zakázat nastavením vlastnosti com.ibm.msg.client.commonservices.log.status na hodnotu OFF. Výchozí hodnota této vlastnosti je ON.

Hodnoty `System.err` a `System.out` mohou být nastaveny na odeslání výstupu protokolu do proudů `System.err` a `System.out` .

Úvod do tříd produktu WebSphere MQ pro platformu JMS, pro programátory

Třídy WebSphere MQ pro systém JMS je poskytovatelem rozhraní JMS dodávaným s produktem WebSphere MQ. Třídy WebSphere MQ pro platformu JMS implementuje rozhraní definovaná v balíku javax.jms a dále poskytuje dvě sady rozšíření rozhraní JMS API. Aplikace prostředí Java Platform, Standard Edition (Java SE) a Java Platform, Enterprise Edition (Java EE) mohou používat třídy WebSphere MQ pro JMS.

Specifikace JMS definuje sadu rozhraní, která mohou aplikace používat k provádění operací systému zpráv. Poslední verzí specifikace je verze 1.1. Balík javax.jms uvádí podrobnosti o rozhraních JMS a poskytovatel platformy JMS implementuje tato rozhraní pro specifický produkt systému zpráv. Třídy WebSphere MQ pro systém JMS je poskytovatelem rozhraní JMS, který implementuje rozhraní JMS pro produkt WebSphere MQ.

Tok logiky v rámci aplikace JMS začíná řetězcem ConnectionFactory a objekty Destination. Aplikace používá objekt ConnectionFactory k vytvoření objektu připojení, který představuje aktivní připojení z aplikace k serveru systému zpráv. Aplikace používá objekt Connection k vytvoření objektu relace, který je jediným kontextem podprocesu pro vytváření a příjem zpráv. Aplikace pak může použít objekt Relace a Cílový objekt k vytvoření objektu MessageProducer , který aplikace používá k odeslání zpráv do zadaného místa určení. Místo určení je buď fronta, nebo téma v systému zasílání zpráv a je zapouzdřeno objektem místa určení. Aplikace může také použít objekt Relace a Cílový objekt k vytvoření objektu MessageConsumer , který aplikace používá k příjmu zpráv, které byly odeslány do zadaného cíle.

Specifikace JMS očekává objekty ConnectionFactory a Destination s objekty, které mají být spravovány. Administrátor vytváří a udržuje spravované objekty v centrálním úložišti a aplikace JMS načte tyto objekty pomocí rozhraní JNDI (Java Naming and Directory Interface). Úložiště spravovaných objektů může být v rozsahu od jednoduchého souboru do adresáře LDAP (Lightweight Directory Access Protocol).

Třídy WebSphere MQ pro platformu JMS podporují použití spravovaných objektů. Aplikace může používat všechny funkce produktu WebSphere MQ , které jsou vystaveny prostřednictvím tříd produktu WebSphere MQ pro službu JMS bez jakýchkoli specifických informací o produktu WebSphere MQ, které jsou pevně naprogramovány do aplikace samotné. Toto uspořádání poskytuje aplikaci se stupněm nezávislosti na základní konfiguraci produktu WebSphere MQ . Aby bylo možné tuto nezávislost dosáhnout, může aplikace pomocí rozhraní JNDI načítat továrny připojení a cíle, které jsou uloženy jako spravované objekty, a používat pouze rozhraní definovaná v balíku javax.jms k provádění operací systému zpráv. Administrátor může pomocí nástroje pro administraci produktu WebSphere MQ JMS nebo produktu IBM WebSphere MQ Explorer vytvářet a spravovat spravované objekty v centrálním úložišti. Aplikační server však obvykle poskytuje vlastní úložiště pro spravované objekty a své vlastní nástroje pro vytváření a údržbu objektů. Aplikace Java EE může proto pomocí rozhraní JNDI načítat spravované objekty buď z úložiště aplikačního serveru, nebo z centrálního úložiště.

Třídy WebSphere MQ pro rozhraní JMS také poskytují rozšíření rozhraní JMS API. Předchozí vydání tříd produktu WebSphere MQ pro službu JMS obsahují rozšíření, která jsou implementována v objektech MQConnectionFactory, MQQueue a MQTopic. Tyto objekty mají vlastnosti a metody, které jsou specifické pro produkt WebSphere MQ. Objekty mohou být spravovanými objekty nebo aplikace může dynamicky

vytvářet objekty za běhu. Toto vydání tříd produktu WebSphere MQ pro službu JMS udržuje tato rozšíření a vy můžete nadále používat, beze změny, všechny aplikace, které tato rozšíření používají. Tato rozšíření jsou známa jako rozšíření *WebSphere MQ JMS*. Všimněte si, že v této sadě dokumentace se objekty, které jsou za běhu dynamicky vytvářeny aplikací, *nepovažují* za administrované objekty.

Kromě rozšíření WebSphere MQ JMS tato verze tříd produktu WebSphere MQ pro platformu JMS poskytuje generičtější sadu rozšíření rozhraní JMS API. Tato rozšíření jsou známá jako *rozšíření IBM JMS*a mají následující obecné cíle:

- Chcete-li zajistit vyšší úroveň konzistence mezi poskytovateli platformy JMS IBM ,
- Chcete-li usnadnit zápis do aplikace mostu mezi dvěma systémy zasílání zpráv IBM
- Chcete-li usnadnit portům aplikaci z jednoho poskytovatele platformy JMS IBM do jiného

Hlavní zaměření těchto rozšíření se týká vytváření a konfigurace továren připojení a míst určení dynamicky za běhu, ale rozšíření také poskytují funkce, které nejsou přímo spojené se systémem zpráv, jako je funkce určování problémů.

Objekt továrny připojení, fronty nebo tématu vytvořený pomocí rozhraní javax.jms nebo sada rozšíření JMS lze adresovat pomocí některého z těchto rozhraní APIs; a lze jej přetypovat na kterékoli z těchto rozhraní. Chcete-li udržet přenositelnost aplikací na nejvyšší úrovni, použijte nejobecnější rozhraní API, které je vhodné pro vaše požadavky.

Aplikace prostředí Java SE i aplikace Java EE mohou používat třídy WebSphere MQ pro JMS. Na platformě Java EE podporují třídy produktu WebSphere MQ pro platformu JMS dva typy komunikace mezi komponentou aplikace a správcem front WebSphere MQ :

Odchozí komunikace
Prostřednictvím rozhraní API služby JMS vytváří komponenta aplikace přímo připojení ke správci front a poté odesílá a přijímá zprávy.

Aplikační komponenta může být například aplikační klient, servlet, stránka JavaServer), objekt EJB (Enterprise Java Bean) nebo objekt typu message-driven bean (MDB). V tomto typu komunikace kontejner aplikačního serveru poskytuje pouze funkce nízké úrovně pro podporu operací systému zpráv, jako je sdružování připojení a správa podprocesů.

Příchozí komunikace
Zpráva, která dorazí do místa určení, je doručena do objektu MDB, který poté zprávu zpracuje.

Aplikace Java EE používají objekty MDB k asynchronnímu zpracování zpráv pomocí objektů MDB. Objekt MDB se chová jako modul listener pro zprávy rozhraní JMS a je implementován metodou onMessage(), která definuje způsob zpracování zprávy. Objekt MDB je implementován v kontejneru EJB aplikačního serveru. Přesný způsob, jakým je objekt MDB konfigurován, závisí na tom, jaký aplikační server používáte, ale informace o konfiguraci musí určovat, ke kterému správci front se má připojit, jak se připojit ke správci front, který cíl má být sledován pro zprávy, a transakční chování objektu MDB. Tyto informace pak budou použity kontejnerem EJB. Je-li doručena zpráva splňující kritéria výběru objektu typu message-driven bean v určeném místě určení, kontejner EJB používá třídy produktu WebSphere MQ pro službu JMS k načtení zprávy ze správce front a poté předá zprávu do objektu MDB voláním metody onMessage().

Třídy IBM WebSphere MQ pro architekturu JMS

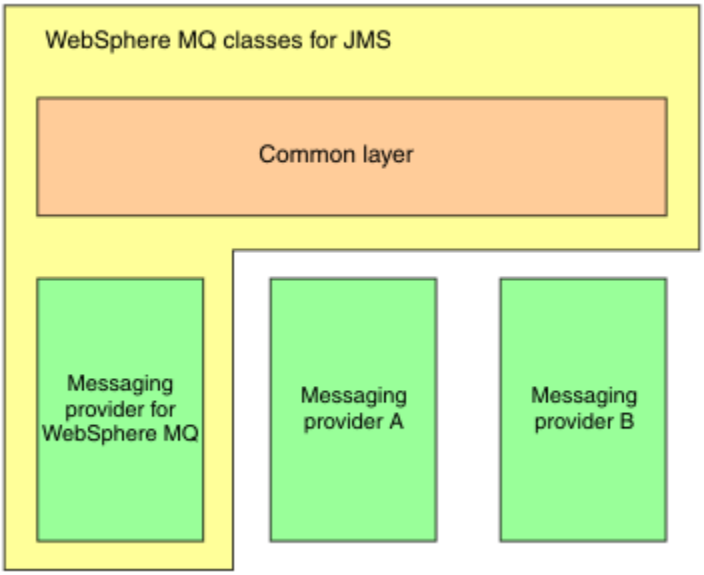
Třídy IBM WebSphere MQ pro službu JMS, jak je dodáváno v produktu IBM WebSphere MQ verze 7.0a v následujících vydáních, obsahují v porovnání s předchozími verzemi několik vylepšení. Některá z těchto vylepšení jsou výsledkem změn implementace tříd produktu IBM WebSphere MQ pro systém JMS a některé z nich jsou výsledkem tříd produktu IBM WebSphere MQ pro službu JMS využívající změny v základní funkci produktu IBM WebSphere MQ .

Následující oddíly shrnují klíčová vylepšení.

Vrstvená architektura

V předchozích vydáních produktu WebSphere MQbyla implementace tříd produktu WebSphere MQ pro službu JMS zcela specifická pro produkt WebSphere MQ. Ostatní produkty IBM , které poskytují systémy zasílání zpráv, obsahovaly také poskytovatele rozhraní JMS, ale tito poskytovatelé JMS mají velmi málo nebo nic společného s implementací tříd produktu WebSphere MQ pro platformu JMS.

V produktu WebSphere MQ V7.0má třídy WebSphere MQ pro JMS vrstvenou architekturu. Nejvyšší vrstva kódu je běžná vrstva, kterou může použít kterýkoli poskytovatel platformy JMS společnosti IBM . Když aplikace volá metodu JMS, provede se veškerá zpracování volání, která není specifická pro systém zasílání zpráv, prováděna běžnou vrstvou, která také poskytuje konzistentní odpověď na volání. Veškeré zpracování volání, které je specifické pro systém zpráv, je delegováno na nižší vrstvu. Obrázek 125 na stránce 772 ukazuje vrstvenou architekturu.



Obrázek 125. Vrstvená architektura pro poskytovatele IBM JMS

Přesun na vrstvenou architekturu má následující cíle:

- Chcete-li zlepšit konzistenci chování různých poskytovatelů IBM JMS,
- Chcete-li usnadnit zápis do aplikace mostu mezi dvěma systémy zasílání zpráv IBM
- Chcete-li usnadnit portům aplikaci z jednoho poskytovatele platformy JMS IBM do jiného

Tato implementace tříd produktu WebSphere MQ pro platformu JMS také zavádí novou sadu rozšíření rozhraní JMS API. Tato rozšíření jsou známá jako *rozšíření IBM JMS*. Hlavní zaměření těchto rozšíření se týká vytváření a konfigurace továren připojení a cílů dynamicky v době běhu programu.

Aplikace používající rozšíření JMS IBM se spustí vytvořením objektu továrny JmsFactory, který určuje jako parametr konstantu, která identifikuje zvolený systém zasílání zpráv. Aplikace používá objekt továrny JmsFactoryk vytváření továren připojení a míst určení, které mají správné specializované třídy pro vybraný systém zasílání zpráv.

Aplikace pak může konfigurovat továrny připojení a cíle nastavením jejich vlastností. Rozšíření IBM JMS poskytují sadu metod pro nastavení vlastností. Tyto metody jsou nezávislé na každém systému zasílání zpráv. Každý datový typ má svou vlastní metodu set a každá vlastnost je identifikována názvem, který je definován jako statický koncový člen třídy WMQConstants. Když aplikace volá jednu z těchto metod, jedním z parametrů v rámci volání je název vlastnosti a druhý parametr je hodnota vlastnosti.

Je-li například produktem WebSphere MQ systém zasílání zpráv, jedním z vlastností továrny připojení je název správce front, k němuž se má připojit. Při použití rozšíření IBM JMS aplikace nastaví název správce front na JUPITER voláním následující metody:

```
JmsConnectionFactory myCF;
...
myCF.setStringProperty(WMQConstants.WMQ_QUEUE_MANAGER, "JUPITER");
```

Aplikace může naproti tomu provádět stejnou funkci voláním následující metody:

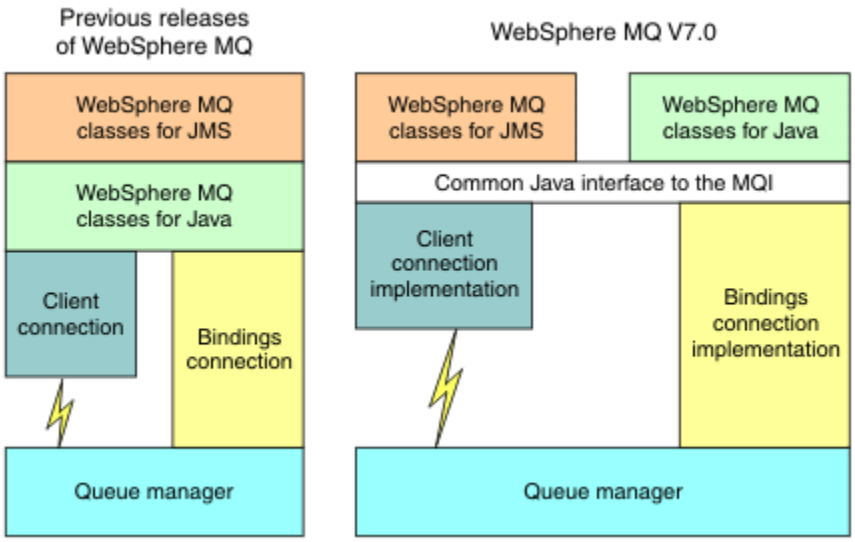
```
MQConnectionFactory myCF;
...
myCF.setQueueManager("JUPITER");
```

Tato metoda je rozšířením produktu WebSphere MQ JMS a je specifická pro produkt WebSphere MQ jako systém zasílání zpráv. Použití této metody proto způsobí, že aplikace bude potenciálně méně snadná k portu jinému poskytovateli platformy JMS IBM .

Vztah mezi třídami WebSphere MQ pro JMS a třídami produktu WebSphere MQ pro prostředí Java

Ve verzích produktu WebSphere MQ, před verzí 7.0, byly třídy WebSphere MQ pro platformu JMS implementovány téměř výhradně jako vrstva kódu v horní části tříd WebSphere MQ pro prostředí Java. Toto uspořádání způsobilo zmatení mezi vývojáři aplikací, protože nastavení polí nebo volání metod ve třídě MQEnvironment může způsobit nechtěné a neočekávané účinky na běhové chování kódu, který je napsán pomocí tříd WebSphere MQ pro JMS. Kromě toho implementace tříd produktu WebSphere MQ pro platformu JMS měla určitá omezení v oblastech, kde rozhraní JMS API není přirozenou součástí tříd produktu WebSphere MQ pro prostředí Java, a tato omezení vedla k problémům s výkonem běhového prostředí.

Od WebSphere MQ V7.0není implementace tříd produktu WebSphere MQ pro platformu JMS již závislá na třídách WebSphere MQ pro jazyk Java. Třídy WebSphere MQ pro jazyk Java a třídy produktu WebSphere MQ pro platformu JMS jsou nyní rovnocenné uzly, které používají společné rozhraní Java pro rozhraní MQI. Toto uspořádání umožňuje větší rozsah optimalizace výkonu a znamená, že nastavení polí nebo volání metod ve třídě MQEnvironment nemá žádný vliv na chování kódu v běhovém prostředí, které je napsáno pomocí tříd WebSphere MQ pro platformu JMS. Obrázek 126 na stránce 773 zobrazuje vztah mezi třídami WebSphere MQ pro platformy JMS a třídami produktu WebSphere MQ pro jazyk Java v předchozích verzích produktu WebSphere MQ a WebSphere MQ V7.0 a dalších verzí.



Obrázek 126. Vztah mezi třídami WebSphere MQ pro JMS a třídami produktu WebSphere MQ pro prostředí Java

Aby byla zachována kompatibilita s dřívějšími verzemi, třídy ukončení kanálu napsané v jazyce Java mohou stále používat třídy WebSphere MQ pro rozhraní Java i v případě, že jsou třídy uživatelské procedury kanálu volány z tříd WebSphere MQ pro JMS. Avšak použití tříd WebSphere MQ pro rozhraní jazyka Java znamená, že vaše aplikace jsou stále závislé na třídách WebSphere MQ pro soubor JAR Java, com.ibm.mq.jar. Pokud ve své cestě ke třídě nechcete soubor com.ibm.mq.jar , můžete místo toho použít novou sadu rozhraní v balíku com.ibm.mq.exits .

Nyní můžete vytvořit a konfigurovat spravované objekty platformy JMS pomocí Průzkumníka produktu WebSphere MQ .

Publikování/odběr zpráv

WebSphere MQ V7.0a následná vydání obsahují funkci vloženého publikování/odběru. Tato funkce nahrazuje publikaci WebSphere MQ Publish/Subscribe, která byla dodána spolu s produktem WebSphere MQ V6.0.

Třídy WebSphere MQ pro aplikace JMS mohou používat vestavěnou funkci publikování/odběru a mohou ji používat místo použití zprostředkovatele událostí WebSphere Event Broker nebo zprostředkovatele zpráv WebSphere Message Broker pro systém zpráv publikování/odběru s produktem WebSphere MQ jako přenosem. Konfigurace tříd produktu WebSphere MQ pro službu JMS pro použití nové funkce je jednodušší než konfigurace tříd produktu WebSphere MQ pro službu JMS pro použití produktu WebSphere MQ Publish/Subscribe, WebSphere Event Broker nebo WebSphere Message Broker. Administrátoři a vývojáři aplikací již nepotřebují spravovat fronty publikování, fronty odběratelů, úložiště odběrů a vyčištění odběratele. Kromě toho mají objekty ConnectionFactory a Topic menší počet vlastností.

Vestavěná funkce publikování/odběru také poskytuje některé další funkce, jako např. zachované publikace a výběr dvou zástupných schémat pro určení rozsahu témat, k jejichž odběru se aplikace chce přihlásit.

Aplikace může stále používat připojení v reálném čase ke zprostředkovateli produktu WebSphere Event Broker nebo zprostředkovatele zpráv WebSphere Message Broker pro systém zpráv publikování/odběru. Tato podpora se nemění.

Aplikace používající produkt WebSphere MQ Publish/Subscribe mohou používat funkci vloženého publikování/odběru beze změny, je-li správce front, k němuž jsou připojeni, upgradován. Vlastnosti nastavené aplikací, které nejsou vyžadovány funkcí vložených publikování/odběru, jsou ignorovány.

Poskytovatel systému zpráv WebSphere MQ

Poskytovatel systému zpráv WebSphere MQ má dva režimy provozu:

- *Normální režim poskytovatele systému zpráv produktu WebSphere MQ*
- *Migrační režim poskytovatele systému zpráv WebSphere MQ*

Normální režim poskytovatele systému zpráv produktu WebSphere MQ používá všechny funkce produktu WebSphere MQ verze 7.0a následné uvolnění správců front k implementaci rozhraní JMS. Tento režim se používá pouze pro připojení ke správci front produktu WebSphere MQ a může se připojit k produktu WebSphere MQ verze 7.0a k následnému uvolnění správců front v režimu klienta nebo vazby. Tento režim je optimalizován pro použití nové funkce produktu WebSphere MQ verze 7.0 a následné funkce uvolnění.

Režim migrace poskytovatele systému zpráv produktu WebSphere MQ je založen na funkci produktu WebSphere MQ verze 6.0 a používá pouze funkce, které byly k dispozici ve správci front WebSphere MQ verze 6.0 pro implementaci platformy JMS. Můžete se připojit k produktu WebSphere MQ verze 7.0 a následující správci front release pomocí režimu migrace poskytovatele systému zpráv produktu WebSphere MQ , ale nelze použít žádné z optimalizací verze 7.0 . Tento režim umožňuje připojení k jednomu z níže uvedených verzí správce front:

1. WebSphere MQ verze 7.0a následný správce front ve vazbách nebo v režimu klienta, ale tento režim používá pouze ty funkce, které byly dostupné pro správce front WebSphere MQ verze 6.0 .
2. WebSphere MQ verze 6.0 nebo starší správce front v režimu klienta

Chcete-li se připojit k produktu WebSphere Event Broker nebo WebSphere Message Broker s použitím produktu WebSphere MQ Enterprise Transport, použijte režim migrace poskytovatele systému zpráv

WebSphere MQ . Používáte-li produkt WebSphere MQ Real-Time Transport, bude automaticky vybrán režim migrace poskytovatele systému zpráv produktu WebSphere MQ , protože jste explicitně vybrali vlastnosti v objektu továrny připojení. Připojení k produktu WebSphere Event Broker nebo WebSphere Message Broker s použitím produktu WebSphere MQ Enterprise Transport odpovídá obecným pravidlům pro výběr režimu popsaným v tématu Pravidla pro výběr režimu poskytovatele systému zpráv produktu WebSphere MQ .

Asynchronní spotřeba zpráv

WebSphere MQ V7.0 a jakékoli následné vydání podporuje asynchronní spotřebu zpráv. Aplikace může registrovat funkci zpětného volání pro místo určení. Je-li do cíle odeslána vhodná zpráva, zavolá produkt WebSphere MQ tuto funkci a předá ji jako parametr. Funkce pak asynchronně zpracuje zprávu. V předchozích verzích produktu WebSphere MQbyla tato funkce k dispozici pouze při použití tříd produktu WebSphere MQ pro službu JMS.

Třídy WebSphere MQ pro platformu JMS byly změněny tak, aby využívaly této nové funkce v produktu WebSphere MQ V7.0 a v dalších verzích. Implementace listenerů zpráv platformy JMS je nyní mnohem přirozenější ve spojení s třídami WebSphere MQa WebSphere MQ pro rozhraní JMS již nemusí vyzývat místo určení, aby zkontrolovala, zda byla do cíle odeslána vhodná zpráva. Výkon listenerů zpráv JMS se v důsledku toho zlepší, zvláště když aplikace používá více modulů listener pro zprávy v rámci relace k monitorování více cílů. Propustnost zpráv se zvyšuje a doba potřebná k doručení zprávy do modulu listener zpráv poté, co se dorazila do místa určení, je zkrácena.

Objekty typu message-driven bean (MDB) mají podobná zlepšení výkonu. Navíc kvůli dalšímu rozšíření funkce produktu WebSphere MQ dochází u více objektů MDB, které spotřebovávají zprávy ze stejného místa určení, za snížené soupeření o zprávy.

Výběr zpráv

S výjimkou výběru zpráv podle identifikátoru zprávy nebo identifikátoru korelace byly všechny výběry zpráv ve verzích produktu WebSphere MQ před verzí 7.0 provedeny třídami produktu WebSphere MQ pro službu JMS. V produktu WebSphere MQ V7.0a ve všech následujících vydáních je veškerá volba zpráv prováděna správcem front.

V důsledku toho se zvýší propustnost zpráv u aplikací, které spotřebovávají zprávy pomocí výběru zpráv. Zlepšení výkonu je větší pro aplikaci, která se připojuje v režimu klienta, protože pouze ty zprávy, které splňují kritéria výběru jsou přeneseny přes síť, a třídy WebSphere MQ pro platformu JMS zpracovávají pouze ty zprávy, které doručí do aplikace.

Sdílení komunikačního připojení

V předchozích verzích produktu WebSphere MQ, pokud byla klientská aplikace produktu WebSphere MQ připojena ke správci front více než jednou pomocí téhož kanálu MQI, je každá instance kanálu MQI vyžadována pro samostatné připojení TCP. V produktu WebSphere MQ V7.0 a libovolné následné verzi může každé připojení ke správci front s použitím téhož kanálu MQI sdílet jedno připojení TCP. Toto uspořádání znamená, že je požadováno méně prostředků sítě a celkový čas potřebný k vytvoření více připojení ke správci front je snížen, zvláště když se používá SSL, protože navázání komunikace SSL probíhá pouze jednou na začátku připojení TCP.

Třídy WebSphere MQ pro platformu JMS využívají toto rozšíření. Pro aplikaci, která se připojuje ke správci front v režimu klienta, třídy WebSphere MQ pro službu JMS mohou vytvořit více než jedno připojení ke správci front s použitím kanálu MQI s názvem, který je určen jako vlastnost objektu ConnectionFactory . Každé z těchto připojení ke správci front může nyní sdílet jedno připojení TCP.

Čtení napřed na připojeních klienta

Pokud aplikace používá připojení klienta pro příjem přechodných zpráv z místa určení, lze cílové místo určení nakonfigurovat tak, aby třídy WebSphere MQ pro platformu JMS používala vyrovnávací paměť k ukládání zpráv, které jsou předmětem zájmu před jejich dodáním do aplikace. Tato optimalizace se nazývá *dopředné čtení* a mohou ji používat aplikace, které spotřebovávají zprávy synchronně voláním

metody receive () a posluchačů zpráv a objektů MDB, které asynchronně spotřebovávají zprávy. Dopředné čtení je zvláště účinné pro místa určení s velkým množstvím zpráv, které je třeba rychle spotřebovat.

Čtení napřed se nevztahuje na trvalé zprávy, protože kdyby byly do vyrovnávací paměti načteny trvalé zprávy, správce front již nebude schopen po selhání obnovit zprávy. Avšak aplikace, která přijímá zprávy z místa určení se směsí trvalých a dočasných zpráv, může stále používat dopředné čtení. Pořadí zpráv je zachováno, ale výhody čtení napřed se vztahují pouze na přechodné zprávy.

Při rozhodování o tom, zda použít čtení napřed, zvažte následující body:

- Pokud aplikace spotřebovává zprávy z místa určení, které je konfigurováno pro dopředné čtení, a aplikace z jakéhokoli důvodu skončí, budou všechny přechodné zprávy, které jsou aktuálně uloženy ve vyrovnávací paměti, vyřazeny.
- Jsou-li všechny následující podmínky pravdivé, zprávy odeslané do fronty v relaci nemusí být přijaty v pořadí, ve kterém byly odeslány:
  - Aplikace používá dva spotřebitele zpráv ve stejné relaci, aby spotřebovávají zprávy z fronty.
  - Každý spotřebitel zpráv používá pro frontu jiný objekt Destination.
  - Libovolný nebo oba objekty Destination jsou konfigurovány pro dopředné čtení.

Odesílání zpráv

Když aplikace odešle zprávy do místa určení, lze cíl nakonfigurovat tak, aby při volání aplikace metody send (), třídy produktu WebSphere MQ pro platformu JMS postoupila zprávu správci front a vrátila řízení zpět do aplikace bez určení, zda správce front zprávu obdržel bezpečně. Třídy WebSphere MQ pro platformu JMS mohou fungovat tímto způsobem pouze pro přechodné zprávy a pro trvalé zprávy odeslané v rámci relace s transakcemi.

Pro trvalé zprávy odeslané v relaci transakce nakonec aplikace určí, zda správce front přijal zprávy bezpečně, když volá potvrzení (). Pro všechny zprávy odeslané v relaci, která není součástí transakce, určuje vlastnost SENDCHECKCOUNT objektu ConnectionFactory počet zpráv, které mají být odeslány před třídami WebSphere MQ pro kontroly JMS, které správce front přijal zprávy bezpečně.

Tato optimalizace má nejvíce výhod 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.

Uživatelské procedury kanálu

Při volání z tříd produktu WebSphere MQ pro platformu JMS se programy uživatelské procedury kanálu napsané v jazycích C nebo C++ chovají stejně, jako když jsou volány z klienta WebSphere MQ MQI. Byl vylepšen výkon tříd uživatelské procedury kanálu, který byl napsán v jazyce Java, a nyní můžete psát uživatelské třídy kanálu pomocí nové sady rozhraní v balíku com.ibm.mq.exits namísto použití rozhraní v třídách WebSphere MQ pro prostředí Java.

Vlastnosti zprávy

Zpráva JMS se skládá ze sady polí záhlaví, sady vlastností a těla, které obsahuje data aplikace. Jako minimum se zpráva WebSphere MQ skládá z deskriptoru zpráv a z dat aplikace.

Když třídy WebSphere MQ pro aplikaci JMS odešlou zprávu JMS, třídy WebSphere MQ pro JMS mapují zprávu JMS na zprávu WebSphere MQ . Některé z polí záhlaví a vlastností záhlaví JMS jsou mapovány na pole v deskriptoru zpráv a některé jsou mapovány na pole v dalších záhlaví WebSphere MQ , které se nazývá záhlaví MQRFH2 . Když třídy WebSphere MQ pro aplikaci JMS obdrží zprávu JMS, třídy WebSphere MQ pro platformu JMS provedou převrácené mapování.

Aplikace, která používá rozhraní MQI k příjmu zpráv ze tříd produktu WebSphere MQ pro aplikaci JMS, musí proto být schopna pracovat se záhlavím MQRFH2 . Pokud aplikace nedokáže zpracovat záhlaví MQRFH2 , může být vlastnost TARGCLIENT objektu Destination nastavena tak, aby produktu WebSphere MQ třídám pro platformu JMS neobsahovala záhlaví MQRFH2 ve zprávách produktu WebSphere

MQ . Nicméně vyloučením záhlaví MQRFH2 se informace obsažené v některých polích záhlaví JMS a vlastnostech ztratí.

Podobně aplikace, která používá rozhraní MQI k odesílání zpráv do tříd produktu WebSphere MQ pro aplikaci JMS, musí obsahovat záhlaví MQRFH2 v každé zprávě. Není-li záhlaví MQRFH2 zahrnuto, třídy WebSphere MQ pro platformu JMS mohou nastavit pouze pole záhlaví JMS a vlastnosti, které lze odvodit z polí v deskriptoru zpráv.

Produkt WebSphere MQ V7.0 poskytuje určitou dodatečnou podporu pro aplikace, které používají rozhraní MQI k přijímání zpráv a odesílání zpráv do tříd produktu WebSphere MQ pro aplikace JMS.

Když aplikace volá příkaz MQGET za účelem přijetí zprávy ze tříd produktu WebSphere MQ pro aplikaci JMS, může aplikace zvolit přijetí zprávy jedním z následujících způsobů:

1. Zpráva je doručena s deskriptorem zpráv, záhlavím MQRFH2 , které obsahuje data odvozená z polí záhlaví a vlastností záhlaví JMS, a data aplikace.
2. Zpráva se dodává s deskriptorem zpráv, daty aplikace a sadou vlastností zprávy.

Ve volbě 2 je každá vlastnost zprávy reprezentována polem záhlaví JMS nebo vlastností, které byly původně mapovány třídami WebSphere MQ pro JMS do pole v záhlaví MQRFH2 . Po volání MQGET může aplikace použít volání MQINQMP k získání hodnot vlastností zprávy. Použití volby 2 místo volby 1 pro přijetí zprávy zjednodušuje logiku aplikace následujícími způsoby:

- Aplikace nemusí analyzovat proměnnou část záhlaví MQRFH2 , která obsahuje pole záhlaví JMS a data vlastností kódovaná ve formátu podobném formátu XML.
- Aplikace nemusí převádět znaková data v proměnné části záhlaví MQRFH2 .

V souladu s tím, než aplikace volá příkaz MQPUT k odeslání zprávy do tříd produktu WebSphere MQ pro aplikaci JMS, může aplikace použít volání MQSETMP k nastavení hodnot vlastností zprávy místo sestavení záhlaví MQRFH2 .

Servisovatelnost

Třídy WebSphere MQ pro platformu JMS obsahují řadu zlepšení týkajících se provozuschopnosti:

- Trasování.

  Třídy WebSphere MQ pro službu JMS obsahují třídu, kterou může aplikace používat k řízení trasování. Aplikace může spustit a zastavit trasování, uvést požadovanou úroveň podrobností v trasování a upravit výstup trasování různými způsoby.
- Protokolování.

  Třídy WebSphere MQ pro platformu JMS udržuje soubor protokolu, který obsahuje zprávy o chybách, které je třeba opravit. Zprávy se zapisují jako prostý text. Třídy WebSphere MQ pro službu JMS obsahují třídu, kterou může aplikace použít k určení umístění souboru protokolu a jeho maximální velikosti.
- First Failure Support Technology (FFST).

  Dojde-li k závažnému selhání, třídy WebSphere MQ pro JMS vygeneruje v souboru FDC zprávu FFST . Sestava FFST obsahuje informace, které může služba IBM Service použít k rychlejšímu diagnostikování problému.
- Informace o verzi.

  Třídy WebSphere MQ pro službu JMS obsahují třídu, kterou může aplikace použít k zadání dotazu na verzi tříd WebSphere MQ pro službu JMS.
- Zprávy výjimek.

  Byly rozšířeny zprávy o výjimce, které poskytují více informací o příčinách chyb a o akcích nezbytných pro opravu chyb.
- Aplikační servery.

  Integrace funkcí služeb WebSphere MQ pro službu JMS s funkcemi serveru WebSphere Application Server byla vylepšena.

Objekt MQC je nahrazen parametrem MQConstants.

Do produktu IBM WebSphere MQ verze 7.0je dodáván nový balík com.ibm.mq.constants. Tento balík obsahuje třídu MQConstants, která implementuje řadu rozhraní. Objekt MQConstants obsahuje definice všech konstant, které se nacházely v rozhraní MQC, a také počet nových konstant. Rozhraní v tomto balíku těsně následují názvy hlavičkového souboru konstant použitých v IBM WebSphere MQ.

Rozhraní CMQC například obsahuje konstantu MQOO_INPUT_SHARED; toto rozhraní a konstantu odpovídají souboru záhlaví `cmqc.h` a konstantě MQOO_INPUT_SHARED.

com.ibm.mq.constants lze použít s oběma třídami IBM WebSphere MQ pro třídy Java a IBM WebSphere MQ pro platformu JMS.

MQC je stále přítomen a má konstanty, které dříve měly. Nicméně pro všechny nové aplikace je třeba použít balík com.ibm.mq.constants .

Zápis tříd produktu WebSphere MQ pro aplikace JMS

Po stručném úvodu k modelu služby JMS poskytuje toto téma podrobné pokyny pro zápis tříd produktu WebSphere MQ pro aplikace JMS.

Model JMS

Model platformy JMS definuje sadu rozhraní, které mohou aplikace Java používat k provádění operací systému zpráv. Třídy WebSphere MQ pro rozhraní JMS jako poskytovatel platformy JMS definují způsob, jakým objekty platformy JMS souvisejí s koncepcemi produktu WebSphere MQ . Specifikace JMS očekává, že určité objekty platformy JMS budou spravovanými objekty.

Specifikace JMS a sada javax.jms definují sadu rozhraní, která mohou aplikace Java používat k provádění operací systému zpráv. Následující seznam shrnuje hlavní rozhraní JMS:

Místo určení
Místo určení je místo, kam aplikace odesílá zprávy, nebo je to zdroj, ze kterého aplikace přijímá zprávy, nebo obojí.

ConnectionFactory
Objekt ConnectionFactory zapouzdřuje sadu vlastností konfigurace pro připojení. Aplikace používá továrnu připojení k vytvoření připojení.

Připojení
Objekt připojení zapouzdřuje aktivní připojení aplikace k serveru systému zpráv. Aplikace používá připojení k vytvoření relací.

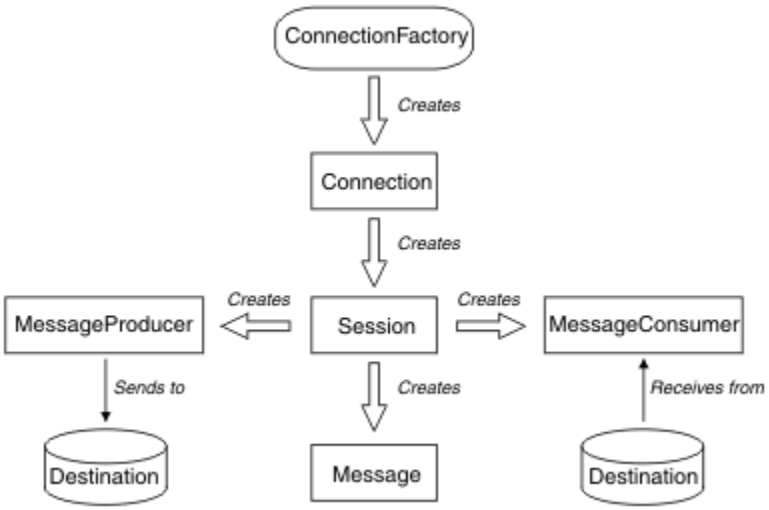
Relace
Relace je jednovláknový kontext pro odesílání a příjem zpráv. Aplikace používá relaci k vytváření zpráv, producentů zpráv a spotřebitelů zpráv. Relace je buď zpracovávána, nebo se nejedná o transakci.

Zpráva
Objekt Message zapouzdřuje zprávu, kterou aplikace odesílá nebo přijímá.

MessageProducer
Aplikace používá producenta zpráv k odesílání zpráv do místa určení.

MessageConsumer
Aplikace používá spotřebitele zpráv k přijetí zpráv odeslaných do místa určení.

Produkt Obrázek 127 na stránce 779 zobrazuje tyto objekty a jejich vztahy.



Obrázek 127. Objekty platformy JMS a jejich vztahy

Místo určení, ConnectionFactorynebo Objekt připojení mohou být použity souběžně různými podprocesy aplikace s více podprocesy, ale objekt Session, MessageProducernebo MessageConsumer nemůže být použit souběžně různými podprocesy. Nejjednodušším způsobem, jak zajistit, aby relace MessageProducernebo MessageConsumer nebyla použita souběžně, je vytvoření samostatného objektu relace pro každý podproces.

Platforma JMS podporuje dva styly systému zpráv:

- Dvoubodový systém zpráv
- Publikování/odběr zpráv

Tyto styly systému zpráv jsou označovány také jako *domény systému zpráv*a v aplikaci můžete kombinovat oba styly systému zpráv. V doméně dvoubodového spojení je cílem fronta a v doméně publikování/odběru je cílem téma.

S verzí JMS před rozhraním JMS 1.1používá programování pro doménu typu point-to-point jednu sadu rozhraní a metod a programování pro doménu publikování/odběru používá jinou sadu. Tyto dvě sady jsou podobné, ale oddělené. S rozhraním JMS 1.1můžete používat společnou sadu rozhraní a metod, které podporují obě domény systému zpráv. Společná rozhraní poskytují nezávislý pohled domény pro každou doménu systému zpráv. Příkaz Tabulka 103 na stránce 779 vypisuje nezávislá rozhraní domény JMS a jejich odpovídající rozhraní specifická pro doménu.

Tabulka 103. Nezávislá a doménová rozhraní domény JMS

Nezávislá rozhraní domény	**Doména specifická pro doménu pro dvoubodovou doménu**	**Rozhraní specifická pro doménu pro doménu publikování/ odběru**
ConnectionFactory	Továrna QueueConnection	Továrna TopicConnection
Připojení	QueueConnection	TopicConnection
Místo určení	Fronta	Téma
Relace	QueueSession	TopicSession
MessageProducer	QueueSender	TopicPublisher
MessageConsumer	QueueReceiver QueueBrowser	TopicSubscriber

Služba JMS 1.1 zachovává všechna rozhraní specifická pro doménu, a tak mohou existující aplikace i nadále používat tato rozhraní. U nových aplikací však zvažte použití nezávislých rozhraní domény.

Ve třídách WebSphere MQ pro platformy JMS se objekty JMS vztahují k konceptům produktu WebSphere MQ následujícími způsoby:

- Objekt připojení má vlastnosti, které jsou odvozeny od vlastností továrny připojení, která byla použita k vytvoření připojení. Tyto vlastnosti řídí způsob, jakým se aplikace připojuje ke správci front. Příklady těchto vlastností jsou název správce front a pro aplikaci, která se připojuje ke správci front v režimu klienta, název hostitele nebo adresa IP systému, v němž je spuštěn správce front.
- Objekt relace zapouzdřuje manipulátor připojení produktu WebSphere MQ , který proto definuje transakční rozsah relace.
- Objekt MessageProducer a objekt MessageConsumer obsahují každý zapouzdřující popisovač objektu WebSphere MQ .

Při použití tříd produktu WebSphere MQ pro systém JMS jsou použita všechna běžná pravidla produktu WebSphere MQ . Všimněte si zejména, že aplikace může odeslat zprávu do vzdálené fronty, ale může přijmout zprávu pouze z fronty, kterou vlastní správce front, ke kterému je aplikace připojena.

Specifikace JMS očekává objekty ConnectionFactory a Destination s objekty, které mají být spravovány. Administrátor vytváří a udržuje spravované objekty v centrálním úložišti a aplikace JMS načte tyto objekty pomocí rozhraní JNDI (Java Naming and Directory Interface).

V třídách WebSphere MQ pro službu JMS je implementace rozhraní Destination Interface abstraktní nadtřída fronty a tématu, takže instance cíle je buď objekt fronty, nebo objekt Topic. Nezávislá rozhraní domény zpracovávají frontu nebo téma jako cíl. Doména systému zpráv pro objekt MessageProducer nebo MessageConsumer je určena podle toho, zda je cílem fronta nebo téma.

V třídách WebSphere MQ pro platformu JMS mohou být objekty následujících typů administrované objekty:

- ConnectionFactory
- Továrna QueueConnection
- Továrna TopicConnection
- Fronta
- Téma
- XAConnectionFactory
- Továrna XAQueueConnection
- Továrna XATopicConnection-továrna

Zprávy JMS

Zprávy JMS se skládají ze záhlaví, vlastností a těla. Platforma JMS definuje pět typů těla zprávy.

Zprávy JMS se skládají z následujících částí:

Header
Všechny zprávy podporují stejnou sadu polí záhlaví. Pole záhlaví obsahují hodnoty, které používají klienti i poskytovatelé k identifikaci a směrování zpráv.

Vlastnosti
Každá zpráva obsahuje vestavěné zařízení pro podporu hodnot vlastností definovaných aplikací. Vlastnosti poskytují účinný mechanizmus pro filtrování zpráv definovaných aplikací.

Tělo
Služba JMS definuje několik typů těla zprávy, které pokrývají většinu stylů systému zpráv, které se aktuálně používají.

Platforma JMS definuje pět typů těla zprávy:

Proud
Proud primitivních hodnot Java. Je vyplněno a postupně čte.

Mapa
Sada dvojic název-hodnota, kde názvy jsou řetězce a hodnoty, jsou primitivní typy Java. K položkám lze přistupovat sekvenčně nebo náhodně podle názvu. Pořadí položek není definováno.

Text
Zpráva obsahující java.lang.String.

Objekt
Zpráva, která obsahuje serializovatelný objekt Java

Bajtů
Proud neinterpretovaného bajtů. Tento typ zprávy je určen pro doslovně kódování textu zprávy tak, aby odpovídal stávajícímu formátu zprávy.

Pole záhlaví JMSCorrelationID se používá k propojení jedné zprávy s jinou. Obvykle propojí zprávu s odpovědí se svou žádající zprávou. JMSCorrelationID může obsahovat ID zprávy specifické pro poskytovatele, řetězec specifický pro aplikaci nebo hodnotu poskytovatele-nativní bajt [] poskytovatele.

Selektory zpráv v JMS

Zprávy mohou obsahovat hodnoty vlastností definované aplikací. Aplikace může používat selektory zpráv k tomu, aby měly zprávy filtru poskytovatele JMS.

Zpráva obsahuje vestavěnou funkci pro podporu hodnot vlastností definovaných aplikací. V důsledku toho poskytuje mechanismus pro přidání polí záhlaví specifického pro aplikaci do zprávy. Vlastnosti umožňují aplikaci pomocí selektorů zpráv vybrat nebo filtrovat zprávy jménem poskytovatele JMS pomocí specifických kritérií aplikace. Vlastnosti definované aplikací musí dodržovat následující pravidla:

- Názvy vlastností musí dodržovat pravidla pro identifikátor selektoru zpráv.
- Hodnoty vlastností mohou být Boolean, byte, short, int, long, float, double a String.
- Předpony názvů JMSX a JMS_ jsou rezervovány.

Hodnoty vlastností jsou nastaveny před odesláním zprávy. Když klient obdrží zprávu, vlastnosti zprávy jsou jen pro čtení. Pokusí-li se klient o nastavení vlastností v tomto bodě, je vyvolána výjimka MessageNotWriteableException . Je-li volána funkce clearProperties , vlastnosti lze nyní číst i zapisovat do pole.

Hodnota vlastnosti může kopírovat hodnotu v těle zprávy. Platforma JMS nedefinuje zásadu pro to, co může být provedeno do vlastnosti. Vývojáři aplikací si však musí být vědomi toho, že poskytovatelé JMS mají pravděpodobně data v těle zprávy účinněji než data ve vlastnostech zprávy. Chcete-li dosáhnout nejlepšího výkonu, aplikace musí používat vlastnosti zprávy pouze v případě, že potřebují upravit záhlaví zprávy. Primárním důvodem pro provedení této činnosti je podpora upraveného výběru zpráv.

Selektor zpráv JMS umožňuje klientovi zadávat zprávy, o které se zajímá, pomocí záhlaví zprávy. Dodány budou pouze zprávy se záhlavími, které odpovídají selektoru.

Selektory zpráv nemohou odkazovat na hodnoty těla zprávy.

Selektor zpráv odpovídá zprávě, je-li selektor vyhodnocen na hodnotu true, když jsou hodnoty pole záhlaví zprávy a hodnoty vlastností nahrazeny pro odpovídající identifikátory v selektoru.

Selektor zpráv je řetězec se syntaxí, která je založena na dílčí sadě syntaxe podmíněného výrazu SQL92 . Pořadí, ve kterém je vyhodnocován selektor zpráv, je zleva doprava v rámci úrovně priority. Chcete-li toto pořadí změnit, můžete použít závorky. Literály předdefinovaných selektorů a názvy operátorů jsou zapsány zde velkými písmeny; avšak nerozlišují se malá a velká písmena.

Selektor může obsahovat:

- literály
  - Řetězcový literál je uzavřený v uvozovkách. Zdvojená uvozovka představuje uvozovku. Příklady jsou 'literal' a 'literal''s '. Podobně jako řetězcové literály Java tyto používají kódování znaků Unicode.
  - Přesný číselný literál je číselná hodnota bez desetinné čárky, jako například 57, -957 a +62. Čísla v rozsahu Java long jsou podporovány.

  - Přibližný numerický literál je číselná hodnota v exponenciální notaci, například 7E3 nebo -57.9E2, nebo číselná hodnota s desítkovým číslem, například 7., -95.7nebo +6.2. Čísla v rozsahu Java double jsou podporovány.
  - Booleovské literály TRUE a FALSE.
- Identifikátory:
  - Identifikátor je neomezená posloupnost písmen Java a číslic Java, přičemž první z nich musí být písmeno Java. Písmeno je libovolný znak, pro který vrací metoda Character.isJavaLetter hodnotu true. To zahrnuje _ a $. Písmeno nebo číslice je libovolný znak, pro který vrací metoda Character.isJavaLetterOrDigit 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.

    Hodnoty JMSMessageID, JMSTimstamp, JMSCorrelationIDa JMSType mohou mít hodnotu null, a pokud ano, jsou považovány za hodnotu NULL.
  - Jakýkoli název začínající na JMSX je název vlastnosti definovaný službou JMS.
  - Všechny názvy začínající řetězcem JMS_ jsou název vlastnosti specifický pro poskytovatele.
  - Jakýkoli název, který nezačíná na JMS, je název vlastnosti specifický pro aplikaci. Existuje-li odkaz na vlastnost, která ve zprávě neexistuje, její hodnota je NULL. Pokud existuje, jeho hodnota je odpovídající hodnota vlastnosti.
- Mezera je stejná, jako je definovaná pro prostředí Java: mezera, vodorovná karta, posuv na novou stránku a koncový znak řádku.
- Výrazy:
  - Selektor je podmíněný výraz. Selektor, který se vyhodnocuje na skutečné shody; selektor, který se vyhodnocuje na hodnotu false nebo neznámý, se neshoduje.
  - Aritmetické výrazy se skládají z sebe, aritmetických operací, identifikátorů (s hodnotou, se kterou se zachází jako s číselným literálem) a numerickými literály.
  - Podmíněné výrazy se skládají z vlastních, porovnávacích operací a logických operací.
- Standardní bracketing () pro nastavení pořadí, ve kterém jsou výrazy vyhodnocovány, je podporován.
- Logické operátory v pořadí priority: NOT, AND, OR.
- Operátory porovnání: =, >, > =, <, < =, < > (nerovná se).
  - Porovnávají se pouze hodnoty stejného typu. Jedna výjimka je, že je platná pro porovnání přesných číselných hodnot a přibližných číselných hodnot. (Požadovaná konverze typu je definována pomocí pravidel pro numerickou propagaci Java.) Pokud dojde k pokusu o porovnání různých typů, je selektor vždy nepravdivý.
  - Porovnání řetězců a logických hodnot je omezeno na hodnoty = a < >. Dva řetězce jsou shodné pouze v případě, že obsahují stejnou posloupnost znaků.
- Aritmetické operátory v pořadí priority:
  - +,-unární.

  - *,/, násobení a dělení.
  - +,-, sčítání a odečítání.
  - Aritmetické operace na hodnotě NULL nejsou podporovány. Pokud se o tyto pokusy pokusí, úplný selektor je vždy nepravdivý.
  - Aritmetické operace musí používat numerickou propagaci Java.
- arithmetic-expr1 [NOT] BETWEEN arithmetic-expr2 a arithmetic-expr3 operátor porovnání:
  - Věk BETWEEN 15 a 19 je ekvivalentní věku > = 15 AND věk < = 19.
  - Věk NEBUDE BETWEEN 15 a 19 je ekvivalentní věku < 15 OR věk > 19.
  - Má-li některý z výrazů operace BETWEEN hodnotu NULL, je hodnota operace false. Má-li některý z výrazů operace NOT BETWEEN hodnotu NULL, je hodnota operace true.
- identifier [NOT] IN (string-literal1, string-literal2, ...) comparison operator where identifier has a String or NULL value.
  - Země IN ("UK", "US", "Francie") platí pro "Spojené království" a "Peru". Je to ekvivalentní výrazu (Country = 'UK ') OR (Country = 'US') OR (Country = 'France ').
  - Země NOT IN ("UK", "US", "Francie") je nepravdivá pro "Spojené království" a platí pro "Peru". Je ekvivalentní výrazu NOT ((Country = 'UK ') OR (Country = 'US') OR (Country = 'France ')).
  - Je-li identifikátor operace IN nebo NOT IN NULL, hodnota operace je neznámá.
- identifier [NOT] LIKE vzor-hodnota [ESCAPE escape-znak] porovnávací operátor, kde identifikátor má hodnotu řetězce. pattern-value is a string literal, where _ stands for any single character and% stands for any sequence of characters (including the empty sequence). Všechny ostatní znaky jsou stojan pro sebe. Volitelný řídicí znak je jednoznakový řetězcový literál, se znakem, který se používá k úniku speciálního významu _ a% ve vzoru-hodnota.
  - telefon LIKE '12%3' je true pro 123 a 12993 a false pro 1234.
  - slovo LIKE 'l_se' je true pro "prohrát" a false pro "loose".
  - podrýhovaný LIKE '_ %' ESCAPE' \ ' je true pro "_foo" a false pro "bar".
  - telefon NOT LIKE '12%3' je false pro 123 a 12993 a true pro 1234.
  - Je-li identifikátor operace LIKE nebo NOT LIKE NULL, hodnota operace je neznámá.
- Identifikátor IS NULL testuje operátor porovnání pro hodnotu pole záhlaví null nebo chybějící hodnotu vlastnosti.
  - prop_name IS NULL.
- Identifikátor testování operátoru IS NOT NULL testuje existenci hodnoty pole záhlaví bez hodnoty null nebo hodnoty vlastnosti.
  - prop_name IS NOT NULL.

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

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

Jak je uvedeno v předchozím seznamu, hodnoty vlastností mohou mít hodnotu NULL. Vyhodnocení výrazů selektoru, které obsahují hodnoty NULL, je definováno sémantikou SQL 92 NULL. Následující seznam uvádí stručný popis těchto sémantik:

- SQL považuje hodnotu NULL za neznámou.
- Porovnání nebo aritmetika s neznámou hodnotou vždy poskytne neznámou hodnotu.
- Operátor IS NULL převede neznámou hodnotu na hodnotu TRUE.
- Operátor IS NOT NULL převede neznámou hodnotu na hodnotu FALSE.

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

Komentáře SQL nejsou podporovány.

Mapování zpráv JMS na zprávy produktu WebSphere MQ

Zprávy produktu WebSphere MQ se skládají z deskriptoru zpráv, volitelného záhlaví MQRFH2 a textu zprávy. Obsah zprávy JMS je částečně mapován a částečně zkopírován do zprávy produktu WebSphere MQ .

Toto téma popisuje, jak struktura zpráv rozhraní JMS popsaná v první části této části je mapována na zprávu produktu WebSphere MQ . Je předmětem zájmu programátorů, kteří chtějí přenášet zprávy mezi aplikacemi JMS a tradičními aplikacemi WebSphere MQ . Je také zajímavé pro lidi, kteří chtějí manipulovat se zprávami přenášenými mezi dvěma aplikacemi JMS, například v implementaci zprostředkovatele zpráv WebSphere Message Broker.

Tato sekce se nepoužije, pokud aplikace používá připojení v reálném čase ke zprostředkovateli. Když aplikace používá připojení v reálném čase, veškerá komunikace se provádí přímo přes TCP/IP; žádné fronty nebo zprávy produktu WebSphere MQ se nepodílejí.

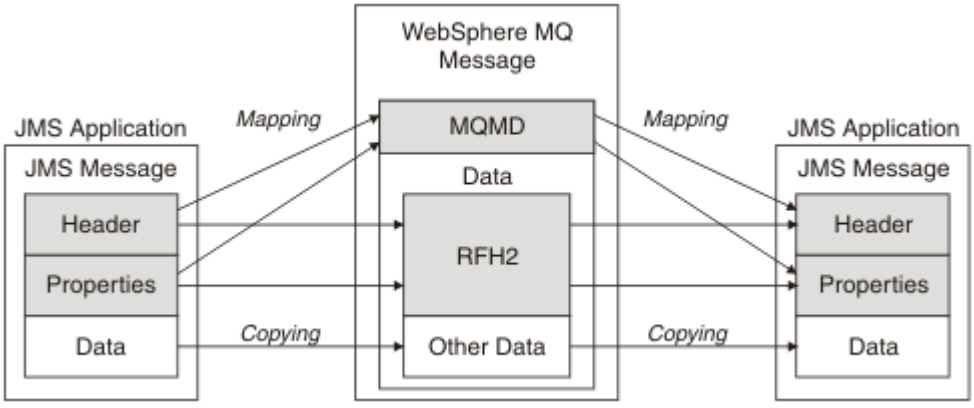
Zprávy produktu WebSphere MQ se skládají ze tří komponent:

- Deskriptor zpráv WebSphere MQ Message Descriptor (MQMD)
- Záhlaví WebSphere MQ MQRFH2 .
- Tělo zprávy.

Hodnota MQRFH2 je volitelná a její zahrnutí do odchozí zprávy se řídí příznakem ve třídě cíle služby JMS. Tento příznak lze nastavit pomocí nástroje pro administraci produktu WebSphere MQ JMS. Protože MQRFH2 obsahuje informace specifické pro JMS, vždy ji zahrňte do zprávy, když odesilatel ví, že přijímající místo určení je aplikací JMS. Obvykle vynechává MQRFH2 při odesílání zprávy přímo do jiné aplikace než JMS. Důvodem je skutečnost, že taková aplikace neočekává MQRFH2 ve své zprávě WebSphere MQ .

Pokud příchozí zpráva nemá záhlaví MQRFH2 , objekt Queue nebo Topic odvozený z pole záhlaví JMSReplyTo zprávy má při výchozím nastavení tento příznak nastaven tak, aby zpráva odpovědi odesílaná do fronty nebo tématu také neodeslala záhlaví MQRFH2 . Toto chování včetně záhlaví MQRFH2 ve zprávě odpovědi můžete vypnout pouze v případě, že původní zpráva má záhlaví MQRFH2 , a to nastavením vlastnosti TARGCLIENTMATCHING továrny na připojení na NO.

Obrázek 128 na stránce 784 ukazuje, jak je struktura zprávy platformy JMS transformována na zprávu WebSphere MQ a znovu se vrátila:



Obrázek 128. Způsob transformace zpráv mezi rozhraním JMS a produktem WebSphere MQ pomocí záhlaví MQRFH2

Struktury jsou transformovány dvěma způsoby:

Mapování
Pokud deskriptor MQMD obsahuje pole, které je ekvivalentní poli JMS, je pole JMS mapováno na pole MQMD. Další pole MQMD jsou vystavena jako vlastnosti JMS, protože aplikace JMS může potřebovat tato pole získat nebo nastavit při komunikaci s aplikací jinou než JMS.

Kopírování
Pokud neexistuje ekvivalentní záhlaví MQMD, je pole záhlaví JMS nebo vlastnost předávána, případně transformována, jako pole v rámci MQRFH2.

Záhlaví a rozhraní JMS MQRFH2 .
Tato kolekce témat popisuje záhlaví MQRFH verze 2, které obsahuje data specifická pro službu JMS, která jsou přidružena k obsahu zprávy. MQRFH2 verze 2 je rozšiřitelné záhlaví a může také nést další informace, které nejsou přímo přidruženy k rozhraní JMS. Tento oddíl se však vztahuje pouze na použití platformy JMS.Úplný popis viz MQRFH2 -Pravidla a formátovací záhlaví 2.

Existují dvě části záhlaví, pevná část a část proměnné.

Pevná část
Pevná část je modelovaná na *standardním* vzoru záhlaví WebSphere MQ a skládá se z následujících polí:

StrucId (MQCHAR4)
Identifikátor struktury.

Musí to být MQRFH_STRUC_ID (hodnota: "RFH ") (počáteční hodnota).

Je také definován parametr MQRFH_STRUC_ID_ARRAY (hodnota: "R","F","H"," ").

Verze (MQLONG)
Číslo verze struktury.

Musí být MQRFH_VERSION_2 (hodnota: 2) (počáteční hodnota).

StrucLength (MQLONG)
Celková délka MQRFH2, včetně datových polí NameValue.

Hodnota nastavená na StrucLength musí být násobkem 4 (data v polích dat NameValuemohou být doplněna mezerami znaků k dosažení tohoto).

Kódování (MQLONG)
Kódování dat.

Kódování jakýchkoli numerických dat v části zprávy za MQRFH2 (další záhlaví nebo data zprávy za tímto záhlavím).

CodedCharSetId (MQLONG)
Identifikátor kódované znakové sady.

Zastupování libovolných znakových dat v části zprávy za MQRFH2 (další záhlaví nebo data zprávy za tímto záhlavím).

Formát (MQCHAR8)
Název formátu.

Název formátu pro část zprávy následující za MQRFH2.

Příznaky (MQLONG)
Příznaky.

MQRFH_NO_FLAGS = 0. Nejsou nastaveny žádné příznaky.

NameValueCCSID (MQLONG)
Identifikátor kódové sady znaků (CCSID) pro řetězce znaků dat NameValueobsažené v tomto záhlaví. Hodnota NameValueData může být kódována ve znakové sadě, která se liší od ostatních znakových řetězců obsažených v záhlaví (StrucID a Format).

Je-li CCSID NameValuedvoubajtový Unicode CCSID (1200, 13488 nebo 17584), je pořadí bajtů Unicode stejné jako pořadí bajtů numerických polí v MQRFH2. (Například, verze, StrucLengtha NameValueCCSID samotné.)

Tabulka 104. Možné hodnoty pro pole NameValueCCSID

Hodnota	Význam
1200	UCS2 open-ended
1208	UTF8
13488	Podmnožina UCS2 2.0
17584	Podmnožina UCS2 2.1 (včetně symbolu eura)

Proměnná část

Proměnná část následuje pevnou část. Proměnná část obsahuje proměnný počet složek MQRFH2 . Každá složka obsahuje proměnný počet prvků nebo vlastností. Vlastnosti související se skupinou složek. Hlavičky MQRFH2 vytvořené rozhraním JMS mohou obsahovat libovolné z následujících složek:

Složka < mcd>

`mcd` obsahuje vlastnosti, které popisují formát zprávy. Například vlastnost `Msd` domény služby zpráv identifikuje zprávu jako typu `JMSTextMessage`, `JMSBytesMessage`, `JMSStreamMessage`, `JMSMapMessage`, `JMSObjectMessage` nebo null.

Složka `mcd` je vždy přítomna ve zprávě JMS obsahující `MQRFH2`.

Vždy se nachází ve zprávě obsahující produkt `MQRFH2` odeslaný z produktu WebSphere Message Broker. Popisuje doménu, formát, typ a sadu zpráv příslušné zprávy.

Tabulka 105. `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 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>

`mqext` obsahuje vlastnosti, které jsou používány pouze serverem WebSphere Application Server. Složka je k dispozici pouze v případě, že aplikace byla nastavena alespoň jednou z definovaných vlastností IBM .

Tabulka 106. 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>

Nepřidávejte své vlastní vlastnosti do složky mqext.

Složka < mqps>

Produkt mqps obsahuje vlastnosti, které jsou používány pouze v rámci publikování/odběru produktu IBM WebSphere MQ. Tato složka je přítomna pouze, když má aplikace nastavenu minimálně jednu z integrovaných vlastností publikování/odběru.

Tabulka 107. 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 108 na stránce 787 zobrazuje úplný seznam názvů vlastností.

Tabulka 108. Složky a vlastnosti MQRFH2 použité službou JMS				
Název pole JMS	**Javovský typ**	**Název složky MQRFH2**	**Název vlastnosti**	**Typ/hodnoty**
JMSDestination	Místo určení	jms	Dst	řetězec
JMSExpiration	long	jms	EXP	i8
JMSPriority.	celé číslo	jms	PRI	i4
JMSDeliveryMode	celé číslo	jms	Dlv	i4

Tabulka 108. Složky a vlastnosti MQRFH2 použité službou JMS (pokračování)				
Název pole JMS	**Javovský typ**	**Název složky MQRFH2**	**Název vlastnosti**	**Typ/hodnoty**
JMSCorrelationID	Řetězec	jms	CID	řetězec
JMSReplyTo	Místo určení	jms	Rto	řetězec
JMSTimestamp	long	jms	Tms	i8
JMSType.	Řetězec	McC	Typ, Nastavit, Fmt	řetězec
JMSXGroupID	Řetězec	jms	GID	řetězec
JMSXGroupSeq	celé číslo	jms	Pořadí	i4
xxx (definovaný uživatelem)	Libovolný	usr	xxx	jakékoli
		McC	MSD	jms_none jms_text jms_bajtů jms_map jms_stream jms_object

NameValueDélka (MQLONG)
Délka v bajtech datového řetězce NameValue, který bezprostředně následuje za tímto polem délky (nezahrnuje jeho vlastní délku).

Data NameValueData (MQCHARn)
Řetězec s jedním znakem, jehož délka v bajtech je dána předchozím polem NameValueLength. Obsahuje složku obsahující posloupnost vlastností. Každá vlastnost je triplet název/typ/hodnota, který je obsažen v prvku XML, jehož název je název složky, a to takto:

```
 <foldername>
triplet1 triplet2 .....   tripletn </foldername>
```

Za uzavírací značkou `</foldername>` může následovat mezery jako doplnění znaků. Každý triplet je zakódován pomocí syntaxe podobné XML:

```
   <name dt='datatype'>value</name>
```

Prvek `dt='datatype'` je volitelný a je vynechán pro mnoho vlastností, protože datový typ je předdefinovaný. Je-li zahrnut, musí být před značku `dt=` zahrnut jeden nebo více mezer.

name
je název vlastnosti; viz Tabulka 108 na stránce 787.

datatype
musí odpovídat, po skládání, jeden z datových typů uvedených v Tabulka 109 na stránce 789.

value
je řetězcová reprezentace hodnoty, která má být předána, s použitím definic v produktu Tabulka 109 na stránce 789.

Hodnota null je zakódována pomocí následující syntaxe:

```
<name dt='datatype' xsi:nil='true'></name>
```

Nepoužívejte `xsi:nil='false'`.

Tabulka 109. Datové typy vlastností	
Datový typ	**Definice**
řetězec	Libovolné pořadí znaků kromě < a &
typ boolean	Znak `0` nebo `1` (`0` = false, `1` = true)
bin.hex	hexadecimální číslice představující oktety
i1	Číslo, vyjádřené pomocí číslic `0..9`, s volitelným znaménkem (bez zlomků nebo exponentu). Musí ležet v rozsahu -128 až 127 včetně
i2	Číslo, vyjádřené pomocí číslic `0..9`, s volitelným znaménkem (bez zlomků nebo exponentu). Musí ležet v rozsahu -32768 až 32767 včetně
i4	Číslo, vyjádřené pomocí číslic `0..9`, s volitelným znaménkem (bez zlomků nebo exponentu). Musí ležet v rozsahu -2147483648 až 2147483647 včetně
i8	Číslo, vyjádřené pomocí číslic `0..9`, s volitelným znaménkem (bez zlomků nebo exponentu). Musí ležet v rozsahu -9223372036854775808 až 922337
celé číslo	Číslo, vyjádřené pomocí číslic `0..9`, s volitelným znaménkem (bez zlomků nebo exponentu). Musí ležet ve stejném rozsahu jako i8. Tuto hodnotu lze použít místo jednoho z typů systému i *, pokud odesilatel nechce přidružit konkrétní přesnost k vlastnosti
r4	Číslo s pohyblivou řádovou čárkou, velikost < = 3.40282347E+38, > = 1.175E-37 vyjádřený pomocí číslic `0..9`, volitelného znaku, volitelných zlomkových číslic, nepovinného exponentu
r8	Číslo s pohyblivou řádovou čárkou, velikost < = 1.7976931348623E+308, > = 2.225E-307 vyjádřená pomocí číslic `0..9`, volitelného znaku, nepovinného zlomkového čísla, volitelného exponentu

Hodnota řetězce může obsahovat mezery. Musíte použít následující řídicí posloupnosti znaků v řetězcové hodnotě:

- `&` pro znak &
- `<` pro znak <

Můžete použít následující řídicí posloupnosti, které však nejsou povinné:

- `>` pro znak >
- `&apos;` pro znak '
- `&quot;` pro znak "

Pole a vlastnosti JMS s odpovídajícími poli MQMD
Tyto tabulky zobrazují pole MQMD ekvivalentní polím záhlaví JMS, vlastnostem platformy JMS a vlastnostem specifickým pro poskytovatele rozhraní JMS.

Tabulka 110 na stránce 789 vypíše pole záhlaví JMS a Tabulka 111 na stránce 790 vypíše vlastnosti platformy JMS, které jsou mapovány přímo na pole MQMD. Tabulka 112 na stránce 790 uvádí seznam vlastností specifických pro poskytovatele a pole MQMD, na které jsou mapovány.

Tabulka 110. Mapování polí záhlaví JMS na pole MQMD			
Pole záhlaví služby JMS	**Javovský typ**	**Pole MQMD**	**Typ C**
JMSDeliveryMode	celé číslo	Trvání	MQLONG
JMSExpiration	long	Vypršení	MQLONG

Tabulka 110. Mapování polí záhlaví JMS na pole MQMD (pokračování)			
Pole záhlaví služby JMS	**Javovský typ**	**Pole MQMD**	**Typ C**
JMSPriority.	celé číslo	Priorita	MQLONG
JMSMessageID	Řetězec	MsgID	MQBYTE24
JMSTimestamp	long	PutDate PutTime	MQCHAR8 MQCHAR8
JMSCorrelationID	Řetězec	CorrelId	MQBYTE24

Tabulka 111. Mapování vlastností JMS na pole MQMD			
Vlastnost JMS	**Javovský 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 112. Mapování vlastností specifických pro poskytovatele JMS na pole MQMD			
Vlastnost specifická pro poskytovatele JMS	**Javovský typ**	**Pole MQMD**	**Typ C**
Výjimka JMS_IBM_Report_Exception	celé číslo	Sestava	MQLONG
JMS_IBM_Report_Expiration.	celé číslo	Sestava	MQLONG
JMS_IBM_Report_COA	celé číslo	Sestava	MQLONG
JMS_IBM_Report_COD.	celé číslo	Sestava	MQLONG
JMS_IBM_Report_PAN.	celé číslo	Sestava	MQLONG
JMS_IBM_Report_NAN.	celé číslo	Sestava	MQLONG
JMS_ID_zprávy_IBM_Report_ID	celé číslo	Sestava	MQLONG
JMS_IBM_Report_Pass_Correl_ID.	celé číslo	Sestava	MQLONG
JMS_IBM_Report_Discard_Msg.	celé číslo	Sestava	MQLONG

Tabulka 112. Mapování vlastností specifických pro poskytovatele JMS na pole MQMD (pokračování)

Vlastnost specifická pro poskytovatele JMS	Javovský typ	Pole MQMD	Typ C
JMS_IBM_MsgType .	celé číslo	MsgType	MQLONG
JMS_IBM_Feedback.	celé číslo	Zpětná vazba	MQLONG
JMS_IBM_Format.	Řetězec	Formát"1" na stránce 791	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 791	MQLONG
JMS_IBM_PutDate .	Řetězec	PutDate	MQCHAR8
JMS_IBM_PutTime .	Řetězec	PutTime	MQCHAR8
JMS_IBM_Last_Msg_In_Group.	typ boolean	MsgFlags	MQLONG

Poznámka:

1. Formát JMS_IBM_Format představuje formát těla zprávy. To může být definováno aplikací, která nastavuje vlastnost JMS_IBM_Format zprávy (všimněte si, že má 8 znaků omezení) nebo může být výchozí pro formát zprávy WebSphere MQ těla zprávy odpovídající typu zprávy JMS. JMS_IBM_Format se mapuje na pole Formát MQMD pouze v případě, že zpráva neobsahuje žádné sekce RFH nebo RFH2 . V typické zprávě je mapován na pole Formát záhlaví RFH2 bezprostředně před tělem zprávy.
2. Hodnota vlastnosti JMS_IBM_Character_Set je řetězcová hodnota, která obsahuje ekvivalent sady znaků jazyka Java pro číselnou hodnotu CodedCharacterSetId . Pole MQMD CodedCharacterSetId je číselná hodnota, která obsahuje ekvivalent řetězce znakové sady Java určený vlastností JMS_IBM_Character_Set.

Mapování polí JMS na pole produktu WebSphere MQ (odchozí zprávy)
Tyto tabulky zobrazují, jak jsou pole záhlaví a vlastností JMS mapována do polí MQMD a MQRFH2 v čase send () nebo publish ().

Tabulka 113 na stránce 792 ukazuje, jak jsou pole záhlaví JMS mapována do polí MQMD/RFH2 v čase send () nebo publish (). Tabulka 114 na stránce 792 ukazuje, jak jsou vlastnosti platformy JMS mapovány do polí MQMD/RFH2 v čase send () nebo publish (). Tabulka 115 na stránce 792 ukazuje, jak jsou vlastnosti specifické pro poskytovatele JMS mapovány na pole MQMD v čase send () nebo publish (),

U polí označených objektem zpráv je přenášená hodnota hodnota zadržená ve zprávě JMS bezprostředně před operací send () nebo publish (). Hodnota ve zprávě JMS je ponechána nezměněná operací.

Pro pole označená Send Method se přiřadí hodnota při provedení akce send () nebo publish () (jakákoli hodnota uchovávané ve zprávě JMS se ignoruje). Hodnota ve zprávě JMS se aktualizuje a zobrazí se použitá hodnota.

Pole označená jako Přijmout nejsou přenesena a jsou ve zprávě ponechána beze změny pomocí funkce send () nebo publish ().

Tabulka 113. Mapování pole odchozí zprávy

Název pole záhlaví služby JMS	Pole MQMD použité pro přenos	Header	Nastavil:
JMSDestination		MQRFH2	Odeslat metodu
JMSDeliveryMode	Trvání	MQRFH2	Odeslat metodu
JMSExpiration	Vypršení	MQRFH2	Odeslat metodu
JMSPriority.	Priorita	MQRFH2	Odeslat metodu
JMSMessageID	MsgID		Odeslat metodu
JMSTimestamp	PutDate/PutTime		Odeslat metodu
JMSCorrelationID	CorrelId	MQRFH2	Objekt zprávy
JMSReplyTo	QMgr ReplyToQ/ReplyToQMgr	MQRFH2	Objekt zprávy
JMSType.		MQRFH2	Objekt zprávy
JMSRedelivered			Pouze příjem

Poznámka:

1. Pole MQMD CodedCharacterSetId je číselná hodnota, která obsahuje ekvivalent řetězce znakové sady Java určený vlastností JMS_IBM_Character_Set.

Tabulka 114. Mapování vlastností odchozí zprávy JMS

Název vlastnosti JMS	Pole MQMD použité pro přenos	Header	Nastavil:
JMSXUserID	UserIdentifier		Odeslat metodu
JMSXAppID	PutApplName		Odeslat metodu
JMSXDeliveryCount			Pouze příjem
JMSXGroupID	GroupId	MQRFH2	Objekt zprávy
JMSXGroupSeq	MsgSeqNumber	MQRFH2	Objekt zprávy

Tabulka 115. Mapování vlastností specifické pro poskytovatele rozhraní JMS odchozí zprávy

Název vlastnosti specifický pro poskytovatele JMS	Pole MQMD použité pro přenos	Header	Nastavil:
Výjimka JMS_IBM_Report_Exception	Sestava		Objekt zprávy
JMS_IBM_Report_Expiration.	Sestava		Objekt zprávy
JMS_IBM_Report_COA/COD.	Sestava		Objekt zprávy
JMS_IBM_Report_NAN/PAN.	Sestava		Objekt zprávy
JMS_ID_zprávy_IBM_Report_ID	Sestava		Objekt zprávy
JMS_IBM_Report_Pass_Correl_ID.	Sestava		Objekt zprávy
JMS_IBM_Report_Discard_Msg.	Sestava		Objekt zprávy
JMS_IBM_MsgType .	MsgType		Objekt zprávy
JMS_IBM_Feedback.	Zpětná vazba		Objekt zprávy

Tabulka 115. Mapování vlastností specifické pro poskytovatele rozhraní JMS odchozí zprávy (pokračování)			
Název vlastnosti specifický pro poskytovatele JMS	**Pole MQMD použité pro přenos**	**Header**	**Nastavil:**
JMS_IBM_Format.	Formát		Objekt zprávy
Typ JMS_IBM_PutAppl	PutApplType		Odeslat metodu
JMS_IBM_Encoding	Kódování		Objekt zprávy
JMS_IBM_Character_Set.	CodedCharacterSetId		Objekt zprávy
JMS_IBM_PutDate .	PutDate		Odeslat metodu
JMS_IBM_PutTime .	PutTime		Odeslat metodu
JMS_IBM_Last_Msg_In_Group.	MsgFlags		Objekt zprávy

Mapování polí záhlaví JMS v operaci send () nebo publish ()
Tyto poznámky se vztahují k mapování polí JMS na odeslání () nebo publish ().

JMSDestination na MQRFH2
Tato hodnota je uložena jako řetězec, který serializuje hlavní charakteristiky cílového objektu, takže přijímající rozhraní JMS může znovu vytvořit ekvivalentní cílový objekt. Pole MQRFH2 je zakódováno jako identifikátor URI (podrobnosti o notaci identifikátoru URI naleznete v části “Uniform resource identifiers (URI)” na stránce 850).

JMSReplyTo na MQMD.ReplyToQ, ReplyToQMgr, MQRFH2 .
Název fronty je zkopírován do MQMD.ReplyToQ a název správce front je zkopírován do polí správce front ReplyToQMgr. Informace o rozšíření místa určení (další užitečné podrobnosti, které jsou uchovány v cílovém objektu) se zkopírují do pole MQRFH2 . Pole MQRFH2 je zakódováno jako identifikátor URI (viz “Uniform resource identifiers (URI)” na stránce 850 . Podrobnosti o notaci identifikátoru URI).

JMSDeliveryMode na MQMD.Persistence
Hodnota JMSDeliveryMode je nastavena metodou send () nebo publish () nebo MessageProducer, pokud objekt Destination Object nepřepíše tento objekt. Hodnota JMSDeliveryMode je mapována na MQMD.Persistence :

- Hodnota JMS PERSISTENT odpovídá MQPER_PERSISTENT.
- Hodnota JMS NON_PERSISTENT odpovídá MQPER_NOT_PERSISTENT.

Není-li vlastnost perzistence MQQueue nastavena na hodnotu WMQConstants.WMQ_PER_QDEF, je hodnota režimu doručení kódována také v rámci struktury MQRFH2.

JMSExpirace do/z MQMD.Expiry, MQRFH2 .
JMSExpiration ukládá čas na vypršení platnosti (součet aktuálního času a doby životnosti), zatímco MQMD ukládá čas k životu. Také hodnota JMSExpiration je v milisekundách, ale MQMD.Expiry je v desetinách sekundy.

- Pokud metoda send () nastavuje neomezenou dobu k životu, MQMD.Expiry je nastaveno na hodnotu MQEI_UNLIMITED a v rámci struktury MQRFH2není zakódována žádná hodnota JMSExpiration.
- Pokud metoda send () nastaví čas životnosti, který je menší než 214748364.7 sekund (asi 7 let), je doba života uložena v MQMD.Expirya doba vypršení platnosti (v milisekundách) je zakódována jako hodnota i8 v MQRFH2.
- Pokud metoda send () nastaví dobu života větší než 214748364.7 sekund, MQMD.Expiry je nastaveno na hodnotu MQEI_UNLIMITED. Doba vypršení platnosti true v milisekundách je zakódována jako hodnota i8 v MQRFH2.

JMSPriority až MQMD.Priority
Přímo namapujte hodnotu JMSPriority (0-9) na hodnotu priority MQMD (0-9). Je-li položka JMSPriority nastavena na jinou než výchozí hodnotu, bude úroveň priority také zakódována v MQRFH2.

JMSMessageID z MQMD.MessageID
Všechny zprávy odeslané z platformy JMS mají jedinečné identifikátory zpráv přiřazené produktem WebSphere MQ. Přiřazená hodnota je vrácena v deskriptoru MQMD.MessageId po volání MQPUT a předává se zpět aplikaci v poli JMSMessageID . WebSphere MQ messageId je 24bajtová binární hodnota, zatímco JMSMessageID je řetězec. Objekt JMSMessageID se skládá z binární hodnoty messageId převedené na posloupnost 48 hexadecimálních znaků s předponou ID znaků:. Služba JMS poskytuje nápovědu, která může být nastavena tak, aby byla zakázána tvorba identifikátorů zpráv. Tento pokyn je ignorován a ve všech případech je přiřazen jedinečný identifikátor. Jakákoli hodnota, která je nastavena do pole JMSMessageId před přepsáním funkce send ().

Vyžadujete-li schopnost určení MQMD.MessageID, můžete to provést s jedním z rozšíření WebSphere MQ JMS popsaných v "Čtení a zápis deskriptoru zpráv ze tříd produktu WebSphere MQ pro aplikaci JMS" na stránce 865.

JMSTimstamp na MQRFH2 .
Během odeslání je pole JMSTimstamp nastaveno v souladu s hodinami prostředí JVM. Tato hodnota je nastavena na MQRFH2. Jakákoli hodnota, která je nastavena do pole JMSTimstamp před přepsáním funkce send (), je přepsána. Viz také vlastnosti JMS_IBM_PutDate a JMS_IBM_PutTime .

JMSType na MQRFH2 .
Tento řetězec je nastaven do pole MQRFH2 mcd.Type . Pokud se nachází ve formátu identifikátoru URI, může také ovlivnit pole mcd.Set a mcd.Fmt . Další informace najdete v tématu "Použití připojení v reálném čase k zprostředkovateli produktu WebSphere Event Broker nebo zprostředkovatele zpráv WebSphere Message Broker" na stránce 892.

JMSCorrelationID na MQMD.CorrelId, MQRFH2 .
JMSCorrelationID může mít jednu z následujících možností:

ID zprávy specifické pro poskytovatele
Jedná se o identifikátor zprávy z dříve odeslané nebo přijaté zprávy, a proto by měl být řetězec 48 hexadecimálních hexadecimálních číslic, který má předponu *ID:* Předpona je odebrána, zbývající znaky jsou převedeny na binární, a pak jsou nastaveny do deskriptoru MQMD.CorrelId . Hodnota CorrelId není zakódována v MQRFH2.

Poskytovatel-nativní bajtová hodnota []
Hodnota je zkopírována do deskriptoru MQMD.CorrelId -vyplněno hodnotami null nebo zkráceno na 24 bajtů, je-li to nutné. Hodnota CorrelId není zakódována v MQRFH2.

Řetězec specifický pro aplikaci
Hodnota je zkopírována do MQRFH2. Prvních 24 bajtů řetězce, ve formátu UTF8 , jsou zapsány do deskriptoru MQMD.CorrelID.

Mapování polí vlastností JMS
Tyto poznámky odkazují na mapování polí vlastností JMS ve zprávách produktu WebSphere MQ .

JMSXUserID z MQMD UserIdentifier
Hodnota JMSXUserID je nastavena na návrat z volání odeslání.

JMSXAppID z názvu MQMD PutAppl
JSMXAppID je nastaven na návrat z volání odeslání.

JMSXGroupID na MQRFH2 (dvoubodový).
Pro zprávy typu point-to-point se zkopíruje JMSXGroupID do pole MQMD GroupID . Pokud se JMSXGroupID spustí s ID prefixu:, je převeden na binární. Jinak je zakódován jako řetězec UTF8 . Hodnota je polstrovaná nebo osekána, je-li to nutné, na délku 24 bajtů. Je nastaven příznak MQMF_MSG_IN_GROUP.

JMSXGroupID na MQRFH2 (publish/subscribe).
U zpráv typu publikování/odběr se JMSXGroupID zkopíruje do struktury MQRFH2 jako řetězec.

JMSXGroupSeq MQMD MsgSeqČíslo (point-to-point)
Pro zprávy typu point-to-point je proměnná JMSXGroupSeq zkopírována do pole Číslo MQMD MsgSeq. Je nastaven příznak MQMF_MSG_IN_GROUP.

JMSXGroupSeq MQMD MsgSeqČíslo (publish/subscribe)
U zpráv typu publikování/odběr se JMSXGroupSeq zkopíruje do MQRFH2 jako i4.

Mapování polí specifických pro poskytovatele JMS
Následující poznámky odkazují na mapování polí specifických pro poskytovatele JMS na zprávy produktu IBM WebSphere MQ .

JMS_IBM_Report_ < název > na sestavu MQMD
Aplikace JMS může nastavit volby sestavy MQMD s použitím následujících vlastností JMS_IBM_Report_XXX. Jediný deskriptor MQMD je mapován na několik vlastností JMS_IBM_Report_XXX. Aplikace musí nastavit hodnotu těchto vlastností na standardní konstanty produktu IBM WebSphere MQ MQRO_ (zahrnuté v com.ibm.mq.MQC). Například, chcete-li požádat o CHSK s úplnými daty, aplikace musí nastavit parametr JMS_IBM_Report_COD na hodnotu CMQC.MQRO_COD_WITH_FULL_DATA.

Výjimka JMS_IBM_Report_Exception

MQRO_EXCEPTION nebo
MQRO_EXCEPTION_WITH_DATA nebo
MQRO_EXCEPTION_WITH_FULL_DATA

JMS_IBM_Report_Expiration.

MQRO_EXPIRATION nebo
MQRO_EXPIRATION_WITH_DATA nebo
MQRO_EXPIRATION_WITH_FULL_DATA

JMS_IBM_Report_COA

MQRO_COA nebo
MQRO_COA_WITH_DATA nebo
MQRO_COA_WITH_FULL_DATA

JMS_IBM_Report_COD.

MQRO_COD nebo
MQRO_COD_WITH_DATA nebo
MQRO_COD_WITH_FULL_DATA

JMS_IBM_Report_PAN.
MQRO_PAN

JMS_IBM_Report_NAN.
MQRO_NAN

JMS_ID_zprávy_IBM_Report_ID
MQRO_PASS_MSG_ID

JMS_IBM_Report_Pass_Correl_ID.
ID_KOLEKCE_MQRO_PASS_RELACE_

JMS_IBM_Report_Discard_Msg.
MQRO_DISCARD_MSG

JMS_IBM_MsgType to MQMD MsgType .
Mapy hodnot přímo na MQMD MsgType. Pokud aplikace nenastavila explicitní hodnotu parametru JMS_IBM_MsgType, použije se výchozí hodnota. Tato výchozí hodnota je určena následujícím způsobem:

- Je-li vlastnost JMSReplyTo nastavena na cíl fronty IBM WebSphere MQ , hodnota MSGType je nastavena na hodnotu MQMT_REQUEST
- Pokud vlastnost JMSReplyTo není nastavena nebo je nastavena na jinou hodnotu než cíl fronty IBM WebSphere MQ , hodnota MsgType je nastavena na hodnotu MQMT_DATAGRAM

JMS_IBM_Feedback k zpětné vazbě MQMD
Mapy hodnot přímo na zpětnou vazbu MQMD.

JMS_IBM_Format na formát MQMD.
Mapy hodnot přímo do formátu MQMD.

JMS_IBM_Encoding na kódování MQMD.
Je-li tato vlastnost nastavena, přepíše číselné kódování cílové fronty nebo tématu.

JMS_IBM_Character_Set to MQMD CodedCharacterSetId .
Je-li nastavena, potlačí tato vlastnost vlastnost kódované znakové sady cílové fronty nebo tématu.

JMS_IBM_PutDate z MQMD PutDate
Hodnota této vlastnosti je nastavena během odesílání přímo z pole PutDate v MQMD. Jakákoli hodnota, která je nastavena do vlastnosti JMS_IBM_PutDate před přepsáním odeslání. Toto pole je řetězec osmi znaků, ve formátu data IBM WebSphere MQ RRRRMMDD. Tuto vlastnost lze použít s vlastností JMS_IBM_PutTime k určení času, kdy byla zpráva vložena do správce front.

JMS_IBM_PutTime z MQMD PutTime .
Hodnota této vlastnosti je nastavena během odeslání přímo z pole PutTime v MQMD. Jakákoli hodnota, která je nastavena do vlastnosti JMS_IBM_PutTime před přepsáním odeslání. Toto pole je řetězec osmi znaků, ve formátu IBM WebSphere MQ Čas HHMMSSTH. Tuto vlastnost lze použít spolu s vlastností JMS_IBM_PutDate k určení času, kdy byla zpráva vložena do správce front.

JMS_IBM_Last_Msg_In_Group-MQMD MsgFlags .
Pro systém zpráv typu point-to-point se tato logická hodnota mapuje na příznak MQMF_LAST_MSG_IN_GROUP v poli MQMD MsgFlags . Obvykle se používá s vlastnostmi JMSXGroupID a JMSXGroupSeq pro označení starší aplikace produktu IBM WebSphere MQ , že tato zpráva je poslední ve skupině. Tato vlastnost je ignorována při publikování/odběru zpráv.

Mapování polí produktu WebSphere MQ na pole JMS (příchozí zprávy)
Tyto tabulky zobrazují, jak jsou pole záhlaví a vlastností JMS mapována do polí MQMD a MQRFH2 v čase get () nebo receive ().

Příkaz Tabulka 116 na stránce 796 zobrazuje, jak jsou pole záhlaví JMS mapována na pole MQMD/MQRFH2 v čase get () nebo receive (). Tabulka 117 na stránce 797 zobrazuje, jak jsou pole vlastností JMS mapována na pole MQMD/MQRFH2 v čase get () nebo receive (). Tabulka 118 na stránce 797 ukazuje, jak jsou mapovány vlastnosti specifické pro poskytovatele JMS.

Tabulka 116. Mapování pole záhlaví JMS příchozí zprávy

Název pole záhlaví služby JMS	**Pole MQMD je načteno z**	**Pole MQRFH2 načteno z**
JMSDestination		jms.Dst nebo mqps.Top"1" na stránce 797
JMSDeliveryMode	Trvání"2" na stránce 797	jms.Dlv"2" na stránce 797
JMSExpiration		jms.Exp
JMSPriority.	Priorita	
JMSMessageID	MsgID	
JMSTimestamp	PutDate"2" na stránce 797 PutTime"2" na stránce 797	jms.Tms"2" na stránce 797
JMSCorrelationID	CorrelId"2" na stránce 797	jms.Cid"2" na stránce 797
JMSReplyTo	ReplyToQ"2" na stránce 797 ReplyTosprávce front"2" na stránce 797	jms.Rto"2" na stránce 797
JMSType.		mcd.Type, mcd.Set, mcd.Fmt
JMSRedelivered	BackoutCount	

Tabulka 116. Mapování pole záhlaví JMS příchozí zprávy (pokračování)

Název pole záhlaví služby JMS	Pole MQMD je načteno z	Pole MQRFH2 načteno z

Poznámka:

1. Je-li nastavena hodnota jms.Dst a mqps.Top , použije se hodnota ve struktuře jms.Dst .
2. Pro vlastnosti, které mohou mít hodnoty načtené ze MQRFH2 nebo MQMD, je-li k dispozici obojí, je použito nastavení v MQRFH2 .
3. Hodnota vlastnosti JMS_IBM_Character_Set je řetězcová hodnota, která obsahuje ekvivalent sady znaků jazyka Java pro číselnou hodnotu CodedCharacterSetId .

Tabulka 117. Mapování vlastností příchozích zpráv

Název vlastnosti JMS	Pole MQMD je načteno z	Pole MQRFH2 načteno z
JMSXUserID	UserIdentifier	
JMSXAppID	PutApplName	
JMSXDeliveryCount	BackoutCount	
JMSXGroupID	GroupId"1" na stránce 797	jms.Gid"1" na stránce 797
JMSXGroupSeq	MsgSeqČíslo"1" na stránce 797	jms.Seq"1" na stránce 797

Poznámka:

1. Pro vlastnosti, které mohou mít hodnoty načtené ze MQRFH2 nebo MQMD, je-li k dispozici obojí, je použito nastavení v MQRFH2 . Vlastnosti jsou nastaveny z hodnot MQMD pouze v případě, že jsou nastaveny příznaky zpráv MQMF_MSG_IN_GROUP nebo MQMF_LAST_MSG_IN_GROUP.

Tabulka 118. Mapování vlastností JMS specifické pro poskytovatele příchozích zpráv

Název vlastnosti JMS	Pole MQMD je načteno z	Pole MQRFH2 načteno z
Výjimka JMS_IBM_Report_Exception	Sestava	
JMS_IBM_Report_Expiration.	Sestava	
JMS_IBM_Report_COA	Sestava	
JMS_IBM_Report_COD.	Sestava	
JMS_IBM_Report_PAN.	Sestava	
JMS_IBM_Report_NAN.	Sestava	
JMS_IBM_Report_ Pass_Msg_ID.	Sestava	
JMS_IBM_Report_Pass_Correl_ID.	Sestava	
JMS_IBM_Report_Discard_Msg.	Sestava	
JMS_IBM_MsgType .	MsgType	
JMS_IBM_Feedback.	Zpětná vazba	
JMS_IBM_Format.	Formát	
Typ JMS_IBM_PutAppl	PutApplType	

Tabulka 118. Mapování vlastností JMS specifické pro poskytovatele příchozích zpráv (pokračování)		
Název vlastnosti JMS	**Pole MQMD je načteno z**	**Pole MQRFH2 načteno z**
JMS_IBM_Encoding [1] .	Kódování	
JMS_IBM_Character_Set [1] .	CodedCharacterSetId	
JMS_IBM_PutDate .	PutDate	
JMS_IBM_PutTime .	PutTime	
JMS_IBM_Last_Msg_In_Group.	MsgFlags	
1. Nastavit pouze v případě, že příchozí zpráva je bajtová zpráva.		

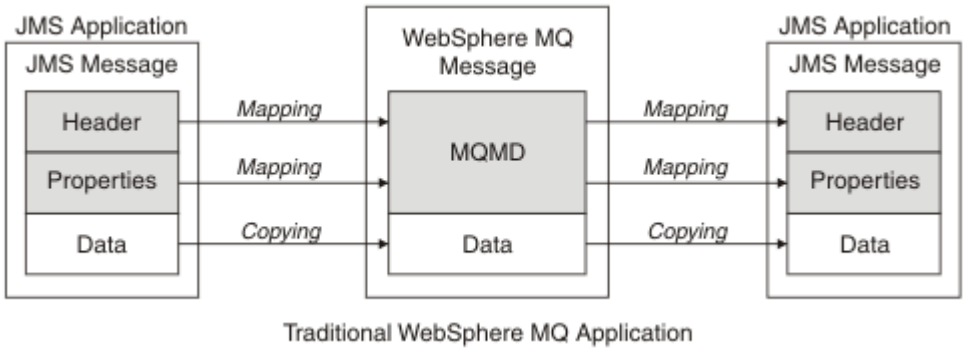
Výměna zpráv mezi aplikací JMS a tradičním produktem WebSphere MQ
Toto téma popisuje, co se stane, když aplikace JMS vyměňuje zprávy s tradiční aplikací produktu WebSphere MQ , která nemůže zpracovat záhlaví MQRFH2 .

. Obrázek 129 na stránce 798 zobrazuje mapování.

Administrátor označuje, že aplikace JMS komunikuje s tradiční aplikací WebSphere MQ tak, že nastaví vlastnost TARGCLIENT cíle na hodnotu *MQ*. To označuje, že se nemá vytvořit žádná hlavička MQRFH2 . Není-li tato akce provedena, přijímající aplikace musí být schopna zpracovat záhlaví MQRFH2 .

Mapování z JMS na MQMD zaměřené na tradiční aplikaci WebSphere MQ je stejné jako mapování z rozhraní JMS na MQMD, které je zaměřeno na aplikaci JMS. Pokud třídy WebSphere MQ pro JMS obdrží zprávu WebSphere MQ s polem MQMD *Format* nastaveným na cokoli jiného než MQFMT_RFH2, data se přijímají z jiné aplikace než JMS. Je-li formát MQFMT_STRING, bude zpráva přijata jako textová zpráva JMS. Jinak se přijme jako bajtová zpráva platformy JMS. Protože zde není žádný MQRFH2, lze obnovit pouze ty vlastnosti JMS, které jsou přeneseny v MQMD.

Pokud třídy WebSphere MQ pro platformu JMS obdrží zprávu, která nemá záhlaví MQRFH2 , vlastnost TARGCLIENT objektu Queue nebo Topic odvozená z pole záhlaví JMSReplyTo zprávy je při výchozím nastavení nastavena na hodnotu MQ . To znamená, že zpráva odpovědi odeslaná do fronty nebo tématu také nemá záhlaví MQRFH2 . Toto chování včetně záhlaví MQRFH2 ve zprávě odpovědi můžete vypnout pouze v případě, že původní zpráva má záhlaví MQRFH2 , a to nastavením vlastnosti TARGCLIENTMATCHING továrny na připojení na NO.



Obrázek 129. Způsob transformace zpráv JMS na zprávy produktu WebSphere 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 Java běžným způsobem.

TextMessage

Řetězec TextMessage je zakódovaný řetězec. U odchozí zprávy je řetězec zakódován ve znakové sadě, která je dána cílovým objektem. Výchozí hodnota je kódování UTF8 (kódování UTF8 začíná prvním znakem zprávy; na začátku není žádné pole délky). Je však možné zadat jakoukoli jinou znakovou sadu podporovanou třídami WebSphere MQ pro JMS. Tyto znakové sady se používají hlavně při odesílání zprávy do jiné aplikace než JMS.

Je-li znaková sada dvoubajtová sada (včetně UTF16), určuje se pořadí bajtů ve specifikaci kódování celých čísel cílového objektu.

Příchozí zpráva je interpretována pomocí znakové sady a kódování, které jsou uvedeny ve zprávě samotné. Tyto specifikace se nacházejí v posledním záhlaví produktu WebSphere MQ (nebo MQMD, pokud nejsou záhlaví). V případě zpráv JMS je posledním záhlavím obvykle hodnota MQRFH2.

BytesMessage

Hodnota BytesMessage je standardně posloupnost bajtů definovaná specifikací JMS 1.0.2 a přidruženou dokumentací Java.

Pro odchozí zprávu, která byla sestavena aplikací samotnou, lze vlastnost kódování cílového objektu použít k potlačení kódování celých čísel a polí s pohyblivou řádovou čárkou obsažených ve zprávě. Můžete například požadovat, aby byly hodnoty s pohyblivou řádovou čárkou uloženy ve formátu S/390 namísto formátu IEEE).

Příchozí zpráva je interpretována s použitím numerického kódování určeného ve zprávě samotné. Tato specifikace se nachází v posledním záhlaví produktu WebSphere MQ (nebo MQMD, pokud nejsou žádná záhlaví). V případě zpráv JMS je posledním záhlavím obvykle hodnota MQRFH2.

Pokud je přijata hodnota BytesMessage a je znovu odeslána bez úpravy, její tělo je přenášen bajt po bajtech, jak bylo přijato. Vlastnost kódování cílového objektu nemá žádný vliv na tělo. Jediná entita podobná řetězci, která může být odeslána explicitně v BytesMessage , je řetězec UTF8 . To je zakódováno ve formátu Java UTF8 a začíná se 2bajtovým polem délky. Vlastnost znakové sady cílového objektu nemá žádný vliv na kódování odchozích BytesMessage. Hodnota znakové sady v příchozí zprávě produktu WebSphere MQ nemá žádný vliv na interpretaci této zprávy jako JMS BytesMessage.

Aplikace jiné než Java pravděpodobně nerozpoznávají kódování jazyka Java UTF8 . Proto pro aplikaci JMS, která má odeslat zprávu BytesMessage obsahující textová data, musí sama aplikace převést své řetězce na pole bajtů a zapsat tato bajtová pole do BytesMessage.

MapMessage

MapMessage je řetězec obsahující triplety XML název/typ/hodnoty kódované jako:

```
<map>
  <elt name="elementname1" dt="datatype1">value1</elt>
  <elt name="elementname2" dt="datatype2">value2</elt>
  ...
</map>
```

kde `datatype` je jeden z datových typů uvedených v Tabulka 109 na stránce 789. Výchozí datový typ je `string`, a proto je atribut `dt="string"` vynechán pro řetězcové prvky.

Znaková sada použitá ke kódování nebo interpretaci řetězce XML, který tvoří tělo mapy zpráv, je určen podle pravidel platných pro textovou zprávu.

Verze produktu WebSphere MQ Classes for JMS starší než verze 5.3 zakódované tělo mapy zpráv v následujícím formátu:

```
<map>
  <elementname1 dt="datatype1">value1</elementname1>
  <elementname2 dt="datatype2">value2</elementname2>
  ...
</map>
```

Verze 5.3 a novější verze tříd produktu WebSphere MQ pro platformu JMS mohou interpretovat buď formát, ale verze tříd produktu WebSphere MQ pro systém JMS starší než verze 5.3 nedokáže interpretovat aktuální formát.

Pokud aplikace potřebuje odeslat mapové zprávy do jiné aplikace, která používá verzi produktu WebSphere MQ pro systém JMS starší než verze 5.3, musí odesílající aplikace volat metodu továrny připojení `setMapNameStyle(WMQConstants.WMQ_MAP_NAME_STYLE_COMPATIBLE)` , aby bylo možné určit, že zprávy mapování jsou odesílány v předchozím formátu. Při výchozím nastavení jsou všechny zprávy mapy odesílány v aktuálním formátu.

StreamMessage
StreamMessage je jako zpráva mapy, ale bez názvů prvků:

```
<stream>
  <elt dt="datatype1">value1</elt>
  <elt dt="datatype2">value2</elt>
  ...
</stream>
```

kde `datatype` je jeden z datových typů uvedených v části Tabulka 109 na stránce 789. Výchozí datový typ je `string`, a proto je atribut `dt="string"` vynechán pro řetězcové prvky.

Znaková sada použitá ke kódování nebo interpretaci řetězce XML, který vytváří tělo StreamMessage , je určen podle pravidel, která se použijí na TextMessage.

Pole MQRFH2.format je nastaveno takto:

MQFMT_NONE
pro ObjectMessage, BytesMessagenebo zprávy bez těla.

ŘETĚZEC MQFMT_STRING
pro TextMessage, StreamMessage, nebo MapMessage.

Převod zpráv platformy JMS

Převod dat zpráv v rozhraní JMS se provádí při odesílání a přijímání zpráv. Produkt WebSphere MQ provádí většinu automatického převodu dat automaticky. Převádí text a číselná data při přenosu zprávy mezi aplikacemi JMS. Text se převede při výměně `JMSTextMessage` mezi aplikací JMS a aplikací WebSphere MQ .

Plánujete-li provádět složitější výměny zpráv, následující témata vás zajímají. Komplexní výměny zpráv zahrnují:

- Přenos netextových zpráv mezi aplikací WebSphere MQ a aplikací JMS.
- Výměna textových dat v bajtovém formátu.
- Probíhá převod textu ve vaší aplikaci.

Data zprávy platformy JMS

Převod dat je nutný pro výměnu textových a číselných dat mezi aplikacemi, a to i mezi dvěma aplikacemi JMS. Interní reprezentace textu a čísel musí být zakódována, aby bylo možné je přenést ve zprávě. Kódování vynutí rozhodnutí o tom, jak jsou čísla a text představeny. Produkt WebSphere MQ spravuje kódování textu a čísel ve zprávách JMS, s výjimkou produktu `JMSObjectMessage`, viz “JMSObjectMessage” na stránce 807. Používá tři atributy zpráv. Uvedené tři atributy jsou `CodedCharacterSetId`, `Encoding`a `Format`.

Tyto tři atributy zpráv jsou obvykle uloženy v záhlaví JMS, `MQRFH2`, pole zprávy JMS. Je-li typ zprávy `MQ`, nikoli `JMS` typu zprávy, jsou atributy uloženy v deskriptoru zpráv, `MQMD`. Atributy se používají pro převod dat zprávy JMS. Data zpráv rozhraní JMS jsou přenášena v části dat zprávy zprávy produktu WebSphere MQ .

Vlastnosti zprávy platformy JMS

Vlastnosti zprávy JMS, jako například `JMS_IBM_CHARACTER_SET`, jsou vyměňovány v části záhlaví `MQRFH2` zprávy JMS, pokud nebyla zpráva odeslána bez `MQRFH2`. Bez `MQRFH2`lze odeslat pouze `JMSTextMessage` a `JMSBytesMessage` . Je-li vlastnost JMS uložena jako vlastnost zprávy produktu WebSphere MQ v deskriptoru zpráv, `MQMD`, je převedena jako součást převodu `MQMD` . Je-li vlastnost služby JMS uložena v produktu `MQRFH2`, je uložena ve znakové sadě určené parametrem

MQRFH2.NameValueCCSID. Je-li zpráva odeslána nebo přijata, jsou vlastnosti zprávy převedeny na jejich interní reprezentaci v prostředí JVM a z jejich interní reprezentace. Převod je nastaven na a ze znakové sady deskriptoru zpráv nebo MQRFH2.NameValueCCSID. Číselná data jsou převedena na text.

Převod zpráv platformy JMS

Následující témata obsahují příklady a úlohy, které jsou užitečné v případě, že plánujete výměnu složitějších zpráv, které vyžadují konverzi.

Přístupy k převodu zpráv JMS
Pro návrháře aplikací JMS je otevřeno řada přístupů pro převod dat. Tyto přístupy se nevylučují; některé aplikace budou pravděpodobně používat kombinaci těchto přístupů. Pokud si aplikace vyměňuje pouze text nebo si vyměňuje zprávy pouze s jinými aplikacemi JMS, obvykle nepovažujete převod dat za normální. Převod dat se provádí automaticky pro vás, v produktu WebSphere MQ.

Můžete se zeptat na několik otázek ohledně toho, jak přistupovat ke konverzi zpráv:

Je třeba přemýšlet o přestavbě zpráv vůbec?
V některých případech, jako je například služba JMS pro přenos zpráv JMS a výměna textových zpráv s programy IBM WebSphere MQ , provádí produkt IBM WebSphere MQ pro vás nezbytné převody automaticky. Možná budete chtít řídit konverzi dat z důvodu výkonu nebo si můžete vyměňovat komplexní zprávy, které mají předdefinovaný formát. V takových případech, jako je tato, musíte rozumět převodu zpráv a přečtěte si následující témata.

Jaký druh konverze je tam?
Existují čtyři hlavní typy převodu, které jsou vysvětleny v následujících sekcích:

1. “Převod dat klienta JMS” na stránce 801
2. “Převod dat aplikace” na stránce 802
3. “Převod dat správce front” na stránce 802
4. “Převod dat kanálu zpráv” na stránce 803

Kde by měl být proveden převod?
Část “Výběr přístupu k převodu zpráv: receiver makes good” na stránce 803popisuje běžný přístup k "přijímači je dobrý". Volba "Receiver makes good" se také používá pro převod dat platformy JMS.

Převod dat klienta JMS

Klient JMS[4]konverze dat je konverze primitiv Java a objektů do bajtů ve zprávě JMS tak, jak je odeslána do místa určení, a konverze zpět, když je přijata. Převod dat klienta JMS používá metody tříd produktu JMSMessage . Metody jsou vypsány podle typu třídy JMSMessage v produktu Tabulka 119 na stránce 804.

Převod na a z vnitřní reprezentace čísel a textu prostředí JVM se provádí pro metody read, get, set a write. Konverze se provede, když se odešle zpráva, a když je některá z metod read nebo get volána na přijatou zprávu.

Kódová stránka a numerické kódování použité k zápisu nebo nastavení obsahu zprávy jsou definovány jako atributy místa určení. Kódu místa určení a číselné kódování lze změnit administrativně. Aplikace může také přepsat cílovou kódovou stránku a kódování nastavením vlastností zprávy, které řídí zápis nebo nastavení obsahu zprávy.

Chcete-li převést kódování čísel při odeslání zprávy produktu JMSBytesMessage na místo určení, které není definováno jako kódování Native , je třeba před odesláním zprávy nastavit vlastnost zprávy JMS_IBM_ENCODING . Pokud sledujete vzor "receiver makes good" , nebo pokud si vyměňujete zprávy mezi aplikacemi JMS, nemusí být aplikace nastavena na JMS_IBM_ENCODING. Ve většině případů můžete opustit vlastnost Encoding jako Native.

[4] "Klient platformy JMS" odkazuje na třídy WebSphere MQ pro rozhraní JMS, které implementují rozhraní JMS, které je spuštěno buď v režimu klienta, nebo vazby.

Pro zprávy JMSStreamMessage, JMSMapMessagea JMSTextMessage se používají vlastnosti identifikátoru znakové sady místa určení. Kódování je při odesílání ignorováno, protože čísla jsou zapsána v textovém formátu. Aplikační program klienta JMS nemusí před odesláním zprávy nastavit JMS_IBM_CHARACTER_SET , pokud se má použít vlastnost cílové znakové sady.

Chcete-li získat data ve zprávě, aplikace volá metodu čtení nebo získání zprávy JMS. Metody odkazují na kódovou stránku a kódování definované v předchozím záhlaví zprávy za účelem vytvoření primitiv Java a objektů správně.

Převod dat klienta JMS vyhovuje potřebám většiny aplikací JMS, které si vyměňují zprávy mezi jedním klientem JMS a druhou. Nekókód explicitního převodu dat nekódujete. Nepoužíváte třídu java.nio.charset.Charset , která se obvykle používá při zápisu textu do souboru. Metody writeString a setString provádí konverzi pro vás.

Další informace o převodu dat klienta JMS naleznete v tématu “Převod a kódování zpráv klienta JMS” na stránce 813.

Převod dat aplikace

Aplikace klienta JMS může provádět explicitní konverzi znakových dat pomocí třídy java.nio.charset.Charset ; viz příklady v Obrázek 132 na stránce 806 a Obrázek 133 na stránce 806 . Řetězcová data se převedou na bajty pomocí metody getBytes a odesílají se jako bajty. Bajty jsou převedeny zpět do textu pomocí konstruktoru String , který přijímá bajtové pole a Charset. Znaková data se konvertují pomocí metod encode a decode Charset . Obvykle je zpráva odeslána nebo přijata jako JMSBytesMessage, protože část zprávy JMSBytesMessage neobsahuje nic jiného než data zapsaná aplikací.[5]bajtů můžete také odesílat a přijímat pomocí produktů JMSStreamMessage, JMSMapMessagenebo JMSObjectMessage.

Neexistují žádné metody Java pro kódování a dekódování bajtů, které obsahují numerická data reprezentovaná v různých formátech kódování. Numerická data jsou kódována a dekódována automaticky pomocí číselných metod čtení a zápisu JMSMessage . Metody čtení a zápisu používají hodnotu atributu JMS_IBM_ENCODING dat zprávy.

Typickým použitím pro převod dat aplikací je to, že klient JMS odešle nebo přijme formátovanou zprávu z jiné aplikace než JMS. Formátovaná zpráva obsahuje text, číselná data a bajtová data uspořádaná podle délky datových polí. Pokud aplikace jiná než JMS neurčila formát zprávy jako "MQSTR ", je zpráva vytvořena jako JMSBytesMessage. Chcete-li přijímat formátovaná data zprávy v produktu JMSBytesMessage , musíte volat posloupnost metod. Metody musí být volány ve stejném pořadí, v jakém byla pole zapsána do zprávy. Jsou-li pole numerická, musíte znát kódování a délku číselných dat. Pokud některá z polí obsahují bajtová nebo textová data, musíte znát délku libovolných bajtových dat ve zprávě. Existují dva způsoby, jak převést naformátovanou zprávu na objekt Java, který se snadno používá.

1. Zkonstruujte třídu Java, která odpovídá záznamu, chcete-li zapouzdřit čtení a zapsat zprávu. Přístup k datům v záznamu je s metodami get a set třídy.
2. Zkonstruujte třídu Java odpovídající záznamu rozšířením třídy com.ibm.mq.headers . Přístup k datům ve třídě je s přístupoovými mechanizmy specifickými pro daný typ formuláře getStringValue(*fieldname*);

Viz “Výměna formátovaného záznamu s aplikací jinou než JMS” na stránce 821.

Převod dat správce front

V produktu WebSphere MQ V7.0může být konverze kódové stránky provedena správcem front, když klientský program JMS obdrží zprávu. Konverze je stejná jako konverze provedená pro program v jazyce C. Programové sady C nastaví MQGMO_CONVERT jako volbu parametru MQGET GetMsgOpts ; viz Obrázek 131 na stránce 806. Správce front provádí převod pro klientský program JMS, který přijímá zprávu, je-li cílová vlastnost WMQ_RECEIVE_CONVERSION nastavena na hodnotu WMQ_RECEIVE_CONVERSION_QMGR, může klientský program JMS také nastavit vlastnost místa určení; viz Obrázek 130 na stránce 803.

[5] Jedna výjimka: Data zapsaná pomocí writeUTF začíná na 2 bajtové délce pole

Před verzí V7.0byly převody vždy prováděny klientem JMS. Převod dat klienta JMS je omezen na převod posloupností čísel a textu typu a délky známé pro klienta JMS. Nelze konvertovat datové struktury; viz “Výměna formátovaného záznamu s aplikací jinou než JMS” na stránce 821. Do verze V7.0, do opravné sady 7.0.1.5, pokud může převod provést správce front, je vždy proveden správcem front. Počínaje verzí 7.0.1.5 se výchozí chování převodu vrací ke stejnému stavu jako V6.0a všechny převody jsou prováděny klientem JMS. Od verze 7.0.1.5nebo 7.0.1.4 s opravou APAR IC72897můžete nastavit novou cílovou volbu, `WMQ_RECEIVE_CONVERSION`, chcete-li řídit, kde je prováděna konverze, a `WMQ_RECEIVE_CCSID`, abyste nastavili cílovou kódovou stránku, viz Obrázek 130 na stránce 803.

```
((MQDestination)destination).setIntProperty(
              WMQConstants.WMQ_RECEIVE_CONVERSION,
              WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
```

Nebo,

```
((MQDestination)destination).setReceiveConversion
             (WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
```

Obrázek 130. Povolit převod dat správce front

Hlavní přínos pro převod správce front je dodáván při výměně zpráv s aplikacemi mimo systém JMS. Je-li definováno pole `Format` ve zprávě a cílová znaková sada nebo kódování se liší od zprávy, správce front provede převod dat pro cílovou aplikaci, pokud ji aplikace požaduje. Správce front převádí data zprávy zformátovaná podle jednoho z předdefinovaných typů zpráv produktu WebSphere MQ , jako je například záhlaví mostu CICS (`MQCIH`). Je-li pole `Format` definováno uživatelem, vyhledá správce front ukončení převodu dat s názvem uvedeným v poli `Format` .

Převod dat správce front se používá k nejlepšímu účinku se vzorem návrhu "receiver makes good" . Odesílající klient JMS nepotřebuje provést převod. Přijímající program, který není rozhraním JMS, se opírá o uživatelskou proceduru převodu, aby se zajistilo, že zpráva bude doručena v požadované kódové stránce a kódování. Při odeslání klienta JMS a jiného příjemce než JMS se tento příklad vztahuje na IBM WebSphere MQ před a post-V7.0. Při použití produktu IBM WebSphere MQ V7.0lze uživatelskou proceduru pro převod volat i pro přijímající program JMS.

Můžete vytvořit uživatelskou proceduru pro převod dat pomocí obslužného programu ukončení převodu dat **`crtmqcvx`**, chcete-li povolit správci front převést vaše vlastní formátovaná data záznamu. Můžete sestavit váš vlastní formát záznamu, použít `com.ibm.mq.headers` pro přístup k němu jako třídu Java a použít vlastní uživatelskou proceduru převodu pro převod. V systému z/OS se obslužný program nazývá **`CSQUCVX`**a v systému IBM i, **`CVTMQMDTA`**. Viz “Výměna formátovaného záznamu s aplikací jinou než JMS” na stránce 821.

Převod dat kanálu zpráv

WebSphere MQ Sender, Server, Cluster-receiver a Cluster-sender channels have a message conversion option, `CONVERT`. Obsah zprávy lze volitelně převést, když se odešle zpráva. Konverze probíhá na odesílajícím konci kanálu. Definice příjemce klastru se používá k automatickému definování odpovídajícího odesílacího kanálu klastru.

Převod dat podle kanálů zpráv se obvykle používá v případě, že není možné použít jiné formy převodu.

Výběr přístupu k převodu zpráv: "receiver makes good"

Obvyklý přístup v návrhu aplikace WebSphere MQ pro převod kódu je "receiver makes good". Volba "Receiver makes good" snižuje počet převodů zpráv. Vyhýbá se také problému neočekávaných chyb kanálu, pokud dojde k selhání převodu zpráv na některém zprostředkujícím správci front během přenosu zprávy. Pravidlo "receiver makes good" je poškozeno pouze v případě, že existuje nějaký důvod, proč nemůže přijímač provádět dobré zpracování. Přijímající platforma možná nemá správnou znakovou sadu, například.

"Receiver makes good" je také dobré obecné pokyny pro klientské aplikace JMS. Avšak v určitých případech může být konverze na správnou znakovou sadu ve zdroji efektivnější. Konverze z interní reprezentace prostředí JVM musí být provedena, když je odeslána zpráva obsahující text nebo numerické typy. Převod na znakovou sadu vyžadovanou příjemcem, pokud příjemce není klientem JMS, může odstranit potřebu provedení konverze pro jiného příjemce než JMS. Je-li příjemce klientem platformy JMS, bude přesto znovu převeden, aby dekódoval data zprávy a vytvořil primitiva a objekty jazyka Java.

Rozdíl mezi aplikacemi klienta JMS a aplikacemi napsanými v jazyce, jako je C, je to, že Java musí provést konverzi dat. Aplikace v jazyce Java musí převádět čísla a text ze své interní reprezentace do kódovaného formátu používaného ve zprávách.

Nastavením místa určení nebo vlastností zprávy můžete nastavit znakovou sadu a kódování používané produktem WebSphere MQ tak, aby kódoval čísla a text ve zprávách. Za normálních okolností byste vynechal znakovou sadu jako `1208` a zakódoval jako `Native`.

WebSphere MQ nekonvertuje bajtová pole. Chcete-li kódovat řetězce a znaková pole do bajtových polí, použijte balík produktu `java.nio.charset` . Parametr `Charset` určuje znakovou sadu použitou k převodu řetězce nebo znakového pole na bajtové pole. bajtové pole lze také dekódovat do řetězce nebo znakového pole pomocí `Charset`. Není dobrým zvykem spoléhat se na `java.nio.charset.Charset.defaultCodePage` při kódování řetězců a znakových polí. Předvolba `Charset` je obvykle `windows-1252` na systému Windowsa `UTF-8` v systému UNIX. `windows-1252` je jednobajtová znaková sada a `UTF-8` je vícebajtová znaková sada.

Obecně nechte cílovou znakovou sadu a vlastnosti kódování při jejich výchozí hodnotě `UTF-8` a `Native` při výměně zpráv s jinými aplikacemi JMS. Pokud si vyměňujete zprávy obsahující čísla nebo text s aplikací JMS, vyberte si jeden z typů zpráv `JMSTextMessage`, `JMSStreamMessage`, `JMSMapMessage`nebo `JMSObjectMessage` , které odpovídají vašemu účelu. Neexistují žádné další úlohy převodu, které by bylo možné provést.

Pokud si vyměňujete zprávy s aplikacemi jiného typu než JMS, které používají formát záznamů, je to složitější. Pokud celý záznam neobsahuje text a může být přenesen jako `JMSTextMessage`, je třeba zakódovat a dekódovat text v aplikaci. Nastavte typ cílové zprávy na `MQ`a použijte `JMSBytesMessage` , abyste se vyvarovali tříd IBM WebSphere MQ pro JMS přidáním dalšího záhlaví a značkování informací do dat zprávy. Použijte metody `JMSBytesMessage` k zápisu čísel a bajtů, a třída `Charset` převádí text do bajtových polí explicitně. Výběr znakové sady může ovlivnit řada faktorů:

- Výkon: Můžete snížit počet převodů transformací textu do znakové sady, která se používá na největším počtu serverů?
- Jednotnost: Přenos všech zpráv ve stejné znakové sadě.
- Richness: Jaké znakové sady mají všechny kódové body, které aplikace musí používat?
- Jednoduchost: jednobajtové znakové sady jsou jednodušší k použití než proměnné délky a vícebajtových znakových sad.

Viz “Výměna formátovaného záznamu s aplikací jinou než JMS” na stránce 821. Příklady převádění zpráv vyměněných s aplikacemi mimo systém JMS.

Příklady

Tabulka typů zpráv a typů převodu

Tabulka 119. Typy zpráv a typy převodu

	Typ převodu			
Typ zprávy	**Text**	**Číselné**	**Jiný**	**Není**
`JMSObjectMessage`				`getObject` `setObject`

Tabulka 119. Typy zpráv a typy převodu (pokračování)

	Typ převodu			
Typ zprávy	**Text**	**Číselné**	**Jiný**	**Není**
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 131. Úsek kódu z `amqsget0.c`

Odesílání a příjem textu v JMSBytesMessage

Kód v produktu Obrázek 132 na stránce 806 odesílá řetězec do pole BytesMessage. Pro zjednodušení příklad odesílá jeden řetězec, pro který je vhodnější `JMSTextMessage` . Chcete-li přijmout textový řetězec v bajtové zprávě obsahující kombinaci typů, musíte znát délku řetězce v bajtech, kterému se říká *TEXT_LENGTH* v produktu Obrázek 133 na stránce 806. I v případě řetězce s pevným počtem znaků může být délka znázornění bajtů delší.

```
BytesMessage bytes = session.createBytesMessage();
String codePage = CCSID.getCodepage(((MQDestination) destination)
                  .getIntProperty(WMQConstants.WMQ_CCSID));
bytes.writeBytes("In the destination code page".getBytes(codePage));
producer.send(bytes);
```

Obrázek 132. Odeslání `String` v `JMSBytesMessage`

```
BytesMessage message = (BytesMessage)consumer.receive();
int TEXT_LENGTH = new Long(message.getBodyLength())).intValue();
byte[] textBytes = new byte[TEXT_LENGTH];
message.readBytes(textBytes, TEXT_LENGTH);
String codePage = message.getStringProperty(WMQConstants.JMS_IBM_CHARACTER_SET);
String textString = new String(textBytes, codePage);
```

Obrázek 133. Příjem `String` z `JMSBytesMessage`

Související pojmy

Převod a kódování zpráv klienta JMS
Jsou vypsány metody, které se používají k provedení konverze a kódování zpráv klienta JMS, s příklady kódu jednotlivých typů konverze.

Převod dat správce front
Převod dat správce front byl vždy k dispozici pro aplikace mimo platformu JMS, které přijímají zprávy od klientů JMS. Vzhledem k tomu, že klienti JMS přijímají zprávy od V7.0, používají také převod dat správce front. Počínaje verzí 7.0.1.5nebo 7.0.1.4 s opravou APAR IC72897je konverze dat správce front volitelná.

Související úlohy

Výměna formátovaného záznamu s aplikací jinou než JMS

Postupujte podle kroků doporučených v této úloze pro návrh a sestavení uživatelské procedury pro převod dat a aplikaci klienta JMS, která může vyměňovat zprávy s aplikací mimo platformu JMS pomocí produktu JMSBytesMessage. Výměnu formátované zprávy s jinou aplikací než JMS se může uskutečnit s voláním ukončení konverze dat nebo bez volání ukončení konverze dat.

Související odkazy

Typy a konverze zpráv rozhraní JMS
Výběr typu zprávy ovlivňuje váš přístup k převodu zpráv. Interakce zprávy o převodu zpráv a typu zprávy jsou popsány pro typy zpráv JMS, JMSObjectMessage, JMSTextMessage, JMSMapMessage, JMSStreamMessagea JMSBytesMessage.

Typy a konverze zpráv rozhraní JMS
Výběr typu zprávy ovlivňuje váš přístup k převodu zpráv. Interakce zprávy o převodu zpráv a typu zprávy jsou popsány pro typy zpráv JMS, JMSObjectMessage, JMSTextMessage, JMSMapMessage, JMSStreamMessagea JMSBytesMessage.

JMSObjectMessage

Produkt JMSObjectMessage obsahuje jeden objekt a všechny objekty, na které odkazuje, které jsou serializovány do bajtového proudu prostředím JVM. Text je serializován do UTF-8a omezen na řetězce nebo pole znaků ne více než 65534 bajtů. Výhodou produktu JMSObjectMessage je, že aplikace nejsou zapojeny do žádných problémů s převodem dat, dokud budou používat pouze metody a atributy objektu. Produkt JMSObjectMessage poskytuje převod dat pro složité objekty bez programátora aplikace s ohledem na to, jak zakódovat objekt ve zprávě. Nevýhodou použití produktu JMSObjectMessage je to, že ji lze vyměňovat pouze s jinými aplikacemi JMS. Vyberete-li jeden z dalších typů zpráv JMS, je možné vyměňovat zprávy JMS s aplikacemi mimo platformu JMS.

“Odesílání a příjem JMSObjectMessage” na stránce 809 zobrazuje objekt String , který je vyměněn ve zprávě.

Klientská aplikace JMS může přijmout JMSObjectMessage pouze ve zprávě, která má tělo ve stylu JMS. Místo určení musí určovat tělo stylu JMS.

JMSTextMessage

JMSTextMessage obsahuje jeden textový řetězec. Je-li odeslána textová zpráva, je text Format nastaven na "MQSTR ", WMQConstants.MQFMT_STRING. CodedCharacterSetId textu je nastaven na identifikátor kódované znakové sady definovaný pro místo určení. Text je zakódován do produktu CodedCharacterSetId produktem WebSphere MQ. Pole CodedCharacterSetId a Format jsou buď nastavena v deskriptoru zpráv, MQMD, nebo do polí JMS v MQRFH2. Pokud je zpráva definována jako mající styl textu zprávy WMQ_MESSAGE_BODY_MQ nebo není zadán styl textu, ale cílové místo určení je WMQ_TARGET_DEST_MQ, pak jsou nastavena pole deskriptoru zpráv. Jinak má zpráva JMS RFH2 a pole jsou nastavena v pevné části MQRFH2.

Aplikace může přepsat identifikátor kódované znakové sady definovaný pro místo určení. Je třeba nastavit vlastnost zprávy JMS_IBM_CHARACTER_SET na identifikátor kódované znakové sady; viz příklad v části “Odesílání a příjem JMSTextmessage” na stránce 809.

Když klient JMS zavolá metodu převodu správce front consumer.receive , je volitelná.
Převod správce front je povolen nastavením vlastnosti místa určení WMQ_RECEIVE_CONVERSION na hodnotu WMQ_RECEIVE_CONVERSION_QMGR. Správce front převede textovou zprávu z JMS_IBM_CHARACTER_SET určeného pro zprávu před přenosem zprávy do klienta JMS. Znaková sada převedené zprávy je 1208, UTF-8, pokud nemá cíl odlišný WMQ_RECEIVE_CCSID. CodedCharacterSetId ve zprávě, která se odkazuje na JMSTextMessage , je aktualizováno na ID cílové znakové sady. Text je dekódován z cílové znakové sady do Unicode pomocí metody getText ; viz příklad v “Odesílání a příjem JMSTextmessage” na stránce 809.

JMSTextMessage může být odeslán v těle zprávy ve stylu MQbez záhlaví JMS MQRFH2 .
Hodnota atributů cíle, WMQ_MESSAGE_BODY a WMQ_TARGET_DEST určují styl těla zprávy, pokud není potlačeno aplikací. Aplikace může přepsat hodnoty nastavené v místě určení

voláním `destination.setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_MQ)` nebo `destination.setTargetClient(WMQConstants.WMQ_TARGET_DEST_MQ)`.

Pokud odešlete produkt `JMSTextMessage` s tělem stylu MQ odesláním do cíle s hodnotou `WMQ_MESSAGE_BODY` nastavenou na hodnotu `WMQ_MESSAGE_BODY_MQ`, nelze ji přijmout jako `JMSTextMessage` ze stejného cíle. Všechny zprávy přijaté z cíle `WMQ_MESSAGE_BODY` nastavené na hodnotu `WMQ_MESSAGE_BODY_MQ` jsou přijaty jako `JMSBytesMessage`. Pokud se pokusíte přijmout zprávu jako `JMSTextMessage` , způsobí to výjimku, `ClassCastException: com.ibm.jms.JMSBytesMessage cannot be cast to javax.jms.TextMessage`.

Poznámka: Text v `JMSBytesMessage` není převáděn klientem JMS. Klient může přijmout text ve zprávě pouze jako bajtové pole. Je-li povolena konverze správce front, je tento text převeden do správce front, ale klient JMS jej musí stále přijímat jako bajtové pole v `JMSBytesMessage`.

Obecně je lepší použít vlastnost `WMQ_TARGET_DEST` k řízení, zda je produkt `JMSTextMessage` odeslán s použitím stylu těla produktu MQ nebo JMS. Poté můžete zprávu přijmout z cíle, který má buď hodnotu `WMQ_TARGET_DEST` nastavenou na hodnotu `WMQ_TARGET_DEST_MQ` nebo `WMQ_TARGET_DEST_JMS`. `WMQ_TARGET_DEST` nemá žádný vliv na přijímač.

JMSMapMessage a JMSStreamMessage

Tyto dva typy zpráv JMS jsou podobné. Primitivní typy můžete číst a zapisovat do zpráv pomocí metod založených na rozhraních `DataInputStream` a `DataOutputStream` , viz “Tabulka typů zpráv a typů převodu” na stránce 812. Podrobnosti jsou popsány v tématu “Převod a kódování zpráv klienta JMS” na stránce 813. Označené primitivum je označeno; viz “Tělo zprávy JMS” na stránce 798.

Číselná data jsou čtena a zapisována do zprávy kódované jako text XML. Neexistuje žádný odkaz na vlastnost místa určení, `JMS_IBM_ENCODING`. Textová data se zpracovaná stejným způsobem jako text v souboru `JMSTextMessage`. Pokud jste se měli podívat na obsah zprávy vytvořený příkladem v produktu Obrázek 138 na stránce 810, všechna data zprávy by byla ve formátu EBCDIC, protože byla odeslána s hodnotou znakové sady 37.

V produktu `JMSMapMessage` nebo `JMSStreamMessage`můžete odeslat více položek.

Jednotlivé položky dat můžete načíst podle názvu z `JMSMapMessage`, nebo podle pozice z `JMSStreamMessage`. Každá položka je dekódována, když je volána metoda get nebo read za použití hodnoty `CodedCharacterSetId` uložené ve zprávě. Pokud metoda použitá k načtení položky vrací jiný typ na typ, který byl odeslán, typ se převede. Pokud typ nelze převést, dojde k výjimce. Podrobnosti najdete v tématu Třída JMSStreamMessage . Příklad v příkladu “Odesílání dat v JMSStreamMessage a JMSMapMessage” na stránce 810 ilustruje konverzi typu a získávání obsahu `JMSMapMessage` mimo pořadí.

Pole `MQRFH2.format` pro `JMSMapMessage` a `JMSStreamMessage` je nastaveno na "`MQSTR  `". Je-li vlastnost cíle `WMQ_RECEIVE_CONVERSION` nastavena na hodnotu `WMQ_RECEIVE_CONVERSION_QMGR`, data zprávy se před odesláním klientovi JMS převede na správce front. `MQRFH2.CodedCharacterSetId` zprávy je `WMQ_RECEIVE_CCSID` cíle. `MQRFH2.Encoding` je `Native`. Má-li proměnná `WMQ_RECEIVE_CONVERSION` hodnotu `WMQ_RECEIVE_CONVERSION_CLIENT_MSG` , hodnota `CodedCharacterSetId` a `Encoding` hodnoty `MQRFH2` je hodnota nastavená odesílatelem.

Aplikace klienta JMS může přijmout `JMSMapMessage` nebo `JMSStreamMessage` pouze ve zprávě, která má tělo ve stylu JMS a z místa určení, které nespecifikuje tělo stylu MQ .

JMSBytesMessage

`JMSBytesMessage` může obsahovat více primitivních typů. Primitivní typy můžete číst a zapisovat do zpráv pomocí metod založených na rozhraních `DataInputStream` a `DataOutputStream` , viz “Tabulka typů zpráv a typů převodu” na stránce 812. Podrobnosti jsou popsány v tématu “Typy a konverze zpráv rozhraní JMS” na stránce 807.

Kódování číselných dat ve zprávě je určováno hodnotou `JMS_IBM_ENCODING` , která je nastavena před zápisem číselných dat do `JMSBytesMessage`. Aplikace může přepsat výchozí kódování `Native` definované pro `JMSBytesMessage` nastavením vlastnosti zprávy `JMS_IBM_ENCODING`.

Textová data lze číst a zapisovat do UTF-8 pomocí `readUTF` a `writeUTF`, nebo v Unicode pomocí metod `readChar` a `writeChar` . K dispozici nejsou žádné metody, které by používaly `CodedCharacterSetId`. Alternativně může klient JMS kódovat a dekódovat text do bajtů pomocí třídy `Charset` . Přenese bajty mezi prostředím JVM a zprávou bez tříd WebSphere MQ pro platformu JMS, která provádí libovolný převod; viz “Odesílání a příjem textu v JMSBytesMessage” na stránce 810.

`JMSBytesMessage` odeslané do aplikace MQ se obvykle odesílá do těla zprávy ve stylu MQbez záhlaví JMS `MQRFH2` . Je-li odeslán do aplikace JMS, styl těla zprávy je obvykle JMS. Hodnota atributů cíle, `WMQ_MESSAGE_BODY` a `WMQ_TARGET_DEST` určují styl těla zprávy, pokud není potlačeno aplikací. Aplikace může přepsat hodnoty nastavené v místě určení voláním `destination.setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_MQ)` nebo `destination.setTargetClient(WMQConstants.WMQ_TARGET_DEST_MQ)`.

Pokud odešlete `JMSBytesMessage` s tělem stylu MQ , můžete přijmout zprávu z cíle, který definuje buď styl těla zprávy produktu MQ , nebo těla zprávy JMS. Pokud odešlete `JMSBytesMessage` s tělem stylu JMS, pak musíte přijmout zprávu z cíle, který definuje styl těla zprávy JMS. Pokud ji neuděláte, bude se s produktem `MQRFH2` zacházet jako s částí dat zprávy uživatele, což nemusí být to, co očekáváte.

Určuje, zda má zpráva styl MQ nebo styl textu zprávy JMS, způsob, jakým je přijat, není ovlivněn nastavením hodnoty `WMQ_TARGET_DEST`.

Zpráva může být později transformována správcem front, pokud je pro data zprávy poskytnuta služba `Format` a je povolen převod dat správce front. Nepoužívejte pole formátu pro cokoli jiného než uvedení formátu dat zprávy, nebo ponechte prázdné, `MQConstants.MQFMT_NONE`

V produktu `JMSBytesMessage`můžete odeslat více položek. Každá číselná položka se převede, když je zpráva odeslána s použitím kódování definovaného pro zprávu.

Jednotlivé položky dat můžete načíst z produktu `JMSBytesMessage`. Volejte metody čtení ve stejném pořadí, v jakém byly volány metody zápisu pro vytvoření zprávy. Každá číselná položka se převede, když se volá zpráva pomocí hodnoty `Encoding` uložené ve zprávě.

Na rozdíl od `JMSMapMessage` a `JMSStreamMessage`, `JMSBytesMessage` obsahuje pouze data zapsaná aplikací. Do dat zprávy nejsou uložena žádná další data, jako např. značky XML použité k definování položek v produktu `JMSMapMessage` a `JMSStreamMessage`. Z tohoto důvodu použijte produkt `JMSBytesMessage` k přenosu zpráv naformátovaných pro jiné aplikace.

Převod mezi `JMSBytesMessage` a `DataInputStream` a `DataOutputStream` je užitečný v některých aplikacích. Kód založený na příkladu, “Čtení a zápis zpráv pomocí DataInputStream a DataOutputStream” na stránce 811, je nezbytný pro použití balíku `com.ibm.mq.header` s platformou JMS.

Příklady

Odesílání a příjem `JMSObjectMessage`

```
ObjectMessage omo = session.createObjectMessage();
omo.setObject(new String("A string"));
producer.send(omo);
...
ObjectMessage omi = (ObjectMessage)consumer.receive();
System.out.println((String)omi.getObject());
...
A string
```

Obrázek 134. Odesílání a příjem `JMSObjectMessage`

Odesílání a příjem `JMSTextmessage`

Textová zpráva nemůže obsahovat text v různých znakových sadách. Příklad ukazuje text v různých znakových sadách, který je odeslán ve dvou různých zprávách.

```
TextMessage tmo = session.createTextMessage();
tmo.setText("Sent in the character set defined for the destination");
producer.send(tmo);
```

Obrázek 135. Odeslat textovou zprávu ve znakové sadě definované v místě určení

```
TextMessage tmo = session.createTextMessage();
tmo.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET, 37);
tmo.setText("Sent in EBCDIC character set 37");
producer.send(tmo);
```

Obrázek 136. Odeslat textovou zprávu v produktu `ccsid` 37

```
TextMessage tmi = (TextMessage)consumer.receive();
System.out.println(tmi.getText());
...
Sent in the character set defined for the destination
```

Obrázek 137. Přijmout textovou zprávu

Odesílání dat v `JMSStreamMessage` a `JMSMapMessage`

```
StreamMessage smo = session.createStreamMessage();
smo.writeString("256");
smo.writeInt(512);
smo.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET, 37);
producer.send(smo);
...
MapMessage mmo = session.createMapMessage();
mmo.setString("First", "256");
mmo.setInt("Second", 512);
mmo.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET, 37);
producer.send(mmo);
...
StreamMessage smi = (StreamMessage)consumer.receive();
System.out.println("Stream: First as float "   + smi.readFloat() +
                           " Second as String " + smi.readString());
...
Stream: First as float: 256.0, Second as String: 512
...
MapMessage mmi = (MapMessage)consumer.receive();
System.out.println("Map: Second as String " + mmi.getString("Second") +
                       " First as double "  + mmi.getDouble("First"));
...
Map: Second as String: 512, First as double: 256.0
```

Obrázek 138. Odeslat data v produktu `JMSStreamMessage` a `JMSMapMessage`

Odesílání a příjem textu v `JMSBytesMessage`

Kód v produktu Obrázek 139 na stránce 811 odesílá řetězec do pole BytesMessage. Pro zjednodušení příklad odesílá jeden řetězec, pro který je vhodnější `JMSTextMessage` . Chcete-li přijmout textový řetězec v bajtové zprávě obsahující kombinaci typů, musíte znát délku řetězce v bajtech, kterému se říká *TEXT_LENGTH* v produktu Obrázek 140 na stránce 811. I v případě řetězce s pevným počtem znaků může být délka znázornění bajtů delší.

```
BytesMessage bytes = session.createBytesMessage();
String codePage = CCSID.getCodepage(((MQDestination) destination)
                  .getIntProperty(WMQConstants.WMQ_CCSID));
bytes.writeBytes("In the destination code page".getBytes(codePage));
producer.send(bytes);
```

Obrázek 139. Odeslání `String` v `JMSBytesMessage`

```
BytesMessage message = (BytesMessage)consumer.receive();
int TEXT_LENGTH = new Long(message.getBodyLength())).intValue();
byte[] textBytes = new byte[TEXT_LENGTH];
message.readBytes(textBytes, TEXT_LENGTH);
String codePage = message.getStringProperty(WMQConstants.JMS_IBM_CHARACTER_SET);
String textString = new String(textBytes, codePage);
```

Obrázek 140. Příjem `String` z `JMSBytesMessage`

Čtení a zápis zpráv pomocí `DataInputStream` a `DataOutputStream`

Kód v produktu Obrázek 141 na stránce 811 vytváří JMSBytesMessage pomocí DataOutputStream.

```
ByteArrayOutputStream bout = new ByteArrayOutputStream();
DataOutputStream dout = new DataOutputStream(bout);
BytesMessage messageOut = prod.session.createBytesMessage();
// messageOut.setIntProperty(WMQConstants.JMS_IBM_ENCODING,
//                           ((MQDestination) (prod.destination)).getIntProperty
//                           (WMQConstants.WMQ_ENCODING));
int ccsidOut = (((MQDestination)prod.destination).getIntProperty(WMQConstants.WMQ_CCSID));
String codePageOut = CCSID.getCodepage(ccsidOut);
dout.writeInt(ccsidOut);
dout.write(codePageOut.getBytes(codePageOut));
messageOut.writeBytes(bout.toByteArray());
producer.send(messageOut);
```

Obrázek 141. Odeslat `JMSBytesMessage` pomocí `DataOutputStream`

Příkaz, který nastaví vlastnost JMS_IBM_ENCODING , je označen jako komentář. Příkaz je platný, pokud se zapisuje přímo do JMSBytesMessage, ale nemá žádný účinek při zápisu do DataOutputStream. Čísla, která jsou zapsána do produktu DataOutputStream , jsou zakódována v kódování Native . Nastavení JMS_IBM_ENCODING nemá žádný efekt.

Kód v produktu Obrázek 142 na stránce 811 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 142. Přijmout `JMSBytesMessage` pomocí `DataInputStream`

Kódová stránka je vytištěna pomocí vlastnosti kódové stránky vstupních dat zprávy, JMS_IBM_CHARACTER_SET. Na vstupu JMS_IBM_CHARACTER_SET je kódová stránka Java a nejedná se o numerický identifikátor kódované znakové sady.

Tabulka typů zpráv a typů převodu

Tabulka 120. Typy zpráv a typy převodu

	Typ převodu			
Typ zprávy	**Text**	**Číselné**	**Jiný**	**Není**
JMSObjectMessage				getObject setObject
JMSTextMessage	getText setText			
JMSBytesMessage	readUTF writeUTF	readDouble readFloat readInt readLong readShort readUnsignedShort writeDouble writeFloat writeInt writeLong writeShort	readBoolean readObject writeBoolean writeObject	readByte readUnsignedByte readBytes readChar writeByte writeBytes writeChar
JMSStreamMessage	readString writeString	readDouble readFloat readInt readLong readShort writeDouble writeFloat writeInt writeLong writeShort	readBoolean readObject writeBoolean writeObject	readByte readBytes readChar writeByte writeBytes writeChar
JMSMapMessage	getString setString	getDouble getFloat getInt getLong getShort setDouble setFloat setInt setLong setShort	getBoolean getObject setBoolean setObject	getByte getBytes readChar setByte setBytes setChar

Související pojmy

Přístupy k převodu zpráv JMS

Pro návrháře aplikací JMS je otevřeno řada přístupů pro převod dat. Tyto přístupy se nevylučují; některé aplikace budou pravděpodobně používat kombinaci těchto přístupů. Pokud si aplikace vyměňuje pouze text nebo si vyměňuje zprávy pouze s jinými aplikacemi JMS, obvykle nepovažujete převod dat za normální. Převod dat se provádí automaticky pro vás, v produktu WebSphere MQ.

Převod a kódování zpráv klienta JMS
Jsou vypsány metody, které se používají k provedení konverze a kódování zpráv klienta JMS, s příklady kódu jednotlivých typů konverze.

Převod dat správce front
Převod dat správce front byl vždy k dispozici pro aplikace mimo platformu JMS, které přijímají zprávy od klientů JMS. Vzhledem k tomu, že klienti JMS přijímají zprávy od V7.0, používají také převod dat správce front. Počínaje verzí 7.0.1.5nebo 7.0.1.4 s opravou APAR IC72897je konverze dat správce front volitelná.

Související úlohy

Výměna formátovaného záznamu s aplikací jinou než JMS
Postupujte podle kroků doporučených v této úloze pro návrh a sestavení uživatelské procedury pro převod dat a aplikaci klienta JMS, která může vyměňovat zprávy s aplikací mimo platformu JMS pomocí produktu `JMSBytesMessage`. Výměnu formátované zprávy s jinou aplikací než JMS se může uskutečnit s voláním ukončení konverze dat nebo bez volání ukončení konverze dat.

Převod a kódování zpráv klienta JMS
Jsou vypsány metody, které se používají k provedení konverze a kódování zpráv klienta JMS, s příklady kódu jednotlivých typů konverze.

Převod a kódování se provádí při čtení nebo zápisu primitiv nebo objektů Java do a ze zpráv JMS. Převod se nazývá konverze dat klienta JMS, aby se odlišil od konverze dat správce front a konverze aplikačních dat. Konverze probíhá striktně, když jsou data čtena nebo zapisována do zprávy JMS. Text je převeden na a z interní 16bitové reprezentace Unicode[6]do znakové sady použité pro text ve zprávách. Numerická data jsou převedena na primitivní číselné typy a primitivní typy Java do kódování definovaného pro zprávu. Zda se provádí konverze a jaký typ převodu se provádí, závisí na typu zprávy JMS a na operaci čtení nebo zápisu.

Tabulka 121 na stránce 813 kategorizuje metody čtení a zápisu pro různé typy zpráv rozhraní JMS podle typu prováděné konverze. Typy převodů jsou popsány v textu následujícím za tabulkou.

Tabulka 121. Typy zpráv a typy převodu

	Typ převodu			
Typ zprávy	**Text**	**Číselné**	**Jiný**	**Není**
`JMSObjectMessage`				`getObject` `setObject`
`JMSTextMessage`	`getText` `setText`			

[6] Některá znázornění Unicode vyžadují více než 16 bitů. Viz odkaz na prostředí Java SE.

Tabulka 121. Typy zpráv a typy převodu (pokračování)				
	Typ převodu			
Typ zprávy	**Text**	**Číselné**	**Jiný**	**Není**
JMSBytesMessage	readUTF writeUTF	readDouble readFloat readInt readLong readShort readUnsignedShort writeDouble writeFloat writeInt writeLong writeShort	readBoolean readObject writeBoolean writeObject	readByte readUnsignedByte readBytes readChar writeByte writeBytes writeChar
JMSStreamMessage	readString writeString	readDouble readFloat readInt readLong readShort writeDouble writeFloat writeInt writeLong writeShort	readBoolean readObject writeBoolean writeObject	readByte readBytes readChar writeByte writeBytes writeChar
JMSMapMessage	getString setString	getDouble getFloat getInt getLong getShort setDouble setFloat setInt setLong setShort	getBoolean getObject setBoolean setObject	getByte getBytes readChar setByte setBytes setChar

Text

Standardní hodnota CodedCharacterSetId pro místo určení je 1208, UTF-8. Standardně je text převeden z Unicode a odeslán jako textový řetězec UTF-8 . Při přijetí se text převede z kódované znakové sady ve zprávě přijaté klientem do Unicode.

Metody setText a writeString převádějí text z kódování Unicode do znakové sady definované pro místo určení. Aplikace může přepsat cílovou znakovou sadu nastavením vlastnosti zprávy JMS_IBM_CHARACTER_SET. JMS_IBM_CHARACTER_SET, při odeslání zprávy musí být číselný identifikátor kódované znakové sady[7].

Úseky kódu v produktu “Odesílání a příjem JMSTextmessage” na stránce 817 posílají dvě zprávy. Jedna se odešle ve znakové sadě definované pro místo určení a druhá ve znakové sadě 37, definované aplikací.

[7] Při přijetí zprávy JMS_IBM_CHARACTER_SET je název kódové stránky jazyka Java produktu Java Charset .

Metody `getText` a `readString` převádějí text ve zprávě ze znakové sady definované ve zprávě do Unicode. Metody používají kódovou stránku definovanou ve vlastnosti zprávy `JMS_IBM_CHARACTER_SET`. Kódová stránka je mapována z produktu `MQRFH2.CodedCharacterSetId` , pokud se nejedná o zprávu typu MQ, která nemá `MQRFH2`. Pokud se jedná o zprávu typu MQtypu -type bez příkazu `MQRFH2`, je kódová stránka mapována z produktu `MQMD.CodedCharacterSetId`.

Úsek kódu v produktu Obrázek 147 na stránce 817 přijímá zprávu, která byla odeslána do cíle. Text ve zprávě je převeden z kódové stránky `IBM037` zpět do Unicode.

Poznámka: Jednoduchým způsobem, jak zkontrolovat, zda je text převeden na kódovanou znakovou sadu 37, je použít produkt WebSphere MQ Explorer. Procházet frontu a zobrazit vlastnosti zprávy před jeho načtením.

Porovnejte úsek kódu v produktu Obrázek 146 na stránce 817 s chybným úsekem kódu v produktu Obrázek 143 na stránce 815. V chybném úseku je textový řetězec převeden dvakrát, jednou aplikací, a znovu produktem WebSphere MQ.

```
TextMessage tmo = session.createTextMessage();
tmo.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET, 37);
tmo.setText(new String("Sent in EBCDIC character set 37".getBytes(CCSID.getCodepage(37))));
producer.send(tmo);
```

Obrázek 143. Nesprávný převod kódové stránky

Metoda `writeUTF` převádí text z kódování Unicode na 1208, `UTF-8`. Textový řetězec je uváděn s délkou 2 bajtové délky. Maximální délka textového řetězce je 65534 bajtů. Metoda `readUTF` čte položku ve zprávě napsanou metodou `writeUTF` . Přečte přesně počet bajtů, které byly zapsány metodou `writeUTF` .

Číselné

Výchozí číselné kódování pro místo určení je `Native`. Kódovací konstanta `Native` pro prostředí Java má hodnotu 273, `x'00000111'`, která je stejná pro všechny platformy. Při přijetí jsou čísla ve zprávě správně transformována do číselných primitiv Java. Transformace používá kódování definované ve zprávě a typ vrácený metodou čtení.

Metoda odeslání převádí čísla, která jsou přidána do zprávy `set` a `write` , do číselného kódování definovaného pro místo určení. Kódování cíle lze přepsat pro zprávu nastavením vlastnosti zprávy `JMS_IBM_ENCODING`; například:

```
message.setIntProperty(WMQConstants.JMS_IBM_ENCODING,
WMQConstants.WMQ_ENCODING_INTEGER_REVERSED);
```

Numerické metody `get` a `read` převádějí čísla ve zprávě z numerického kódování definovaného ve zprávě. Přemění čísla na typ, který je určen metodou `read` nebo `get` ; viz vlastnost ENCODING . Metody používají kódování definované v produktu `JMS_IBM_ENCODING`. Kódování je mapováno z `MQRFH2.Encoding` , pokud se nejedná o zprávu typu MQ, která nemá `MQRFH2`. Pokud se jedná o zprávu typu MQtypu -type bez příkazu `MQRFH2`, pak metody používají kódování definované v produktu `MQMD.Encoding`.

Příklad v produktu Obrázek 148 na stránce 817 zobrazuje aplikaci kódování čísla v cílovém formátu a odeslání je v souboru `JMSStreamMessage`. Porovnejte příklad v produktu Obrázek 148 na stránce 817 s příkladem v produktu Obrázek 149 na stránce 818. Rozdíl je v tom, že `JMS_IBM_ENCODING` musí být nastaveno v `JMSBytesMessage`.

Poznámka: Jednoduchým způsobem, jak zkontrolovat, zda je číslo zakódováno správně, je použití Průzkumníka produktu WebSphere MQ . Procházejte frontu a zobrazte vlastnosti zprávy předtím, než je spotřebována.

Jiný

Metody boolean zakóduje true a false jako x'01' a x'00' v JMSByteMessage, JMSStreamMessagea JMSMapMessage.

Metody UTF zakóduje a dekóduje Unicode do textových řetězců UTF-8 . Řetězce jsou omezeny na méně než 65536 znaků a jsou jim uvedeny 2 bajtové pole délky.

Primitivní typy objektů jsou zalamování objektů jako objekty. Číselné a textové typy jsou kódovány nebo převedeny, jako kdyby byly primitivní typy přečteny nebo zapsány pomocí číselných a textových metod.

Není

Metody readByte, readBytes, readUnsignedByte, writeBytea writeBytes získají nebo vloží jednotlivé bajty nebo pole bajtů mezi aplikací a zprávou bez konverze. Metody readChar a writeChar mohou používat a vkládat 2 bajtové znaky Unicode mezi aplikací a zprávou bez konverze.

Pomocí metod readBytes a writeBytes může aplikace provádět svůj vlastní převod kódových bodů, jako je tomu v části “Odesílání a příjem textu v JMSBytesMessage” na stránce 818.

Produkt WebSphere MQ neprovede v klientovi žádnou konverzi kódové stránky, protože se jedná o zprávu JMSBytesMessage, a protože jsou použity metody readBytes a writeBytes . Nicméně, pokud bajty představují text, ujistěte se, že kódová stránka použitá aplikací odpovídá kódované znakové sadě cíle. Zpráva může být znovu převedena pomocí uživatelské procedury pro převod správce front. Jinou možností je to, že přijímající klientský program JMS může podle konvence konvertovat libovolná bajtová pole reprezentující text ve zprávě do řetězců nebo znaků pomocí vlastnosti JMS_IBM_CHARACTER_SET ve zprávě.

V tomto příkladu klient používá cílovou kódovanou znakovou sadu pro svůj převod:

```
bytes.writeBytes("In the destination code page".getBytes(
      CCSID.getCodepage(((MQDestination) destination)
           .getIntProperty(WMQConstants.WMQ_CCSID))));
```

Případně může klient zvolit kódovou stránku a poté nastavit příslušnou kódovanou znakovou sadu ve vlastnosti JMS_IBM_CHARACTER_SET dané zprávy. Třídy WebSphere MQ pro prostředí Java používají JMS_IBM_CHARACTER_SET pro nastavení pole CodedCharacterSetId ve vlastnostech JMS v produktu MQRFH2nebo v deskriptoru zpráv MQMD:

```
String codePage = CCSID.getCodepage(37);
message.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET, codePage);[8]
```

Pokud je bajtové pole zapsáno do JMSStringMessage nebo JMSMapMessage, třídy WebSphere MQ pro JMS neprovedou konverzi dat, protože bajty jsou zapsány jako hexadecimální data, nikoli jako text v JMSStringMessage a JMSMapMessage.

Pokud bajty představují znaky ve vaší aplikaci, musíte vzít v úvahu, jaké kódové body chcete číst a zapisovat do zprávy. Kód v souboru Obrázek 144 na stránce 817 se řídí konvencemi použití cílové kódové znakové sady. Pokud vytvoříte řetězec pomocí výchozí znakové sady pro prostředí JVM, bude obsah bajtů záviset na platformě. Prostředí JVM v systému Windows má obvykle výchozí hodnotu Charset z windows-1252a UNIX, UTF-8. Interchange mezi systémy Windows a UNIX vyžaduje, abyste vybrali explicitní kódovou stránku pro výměnu textu v bajtech.

[8] SetStringProperty(WMQConstants.JMS_IBM_CHARACTER_SET, codePage) currently accepts only numeric character set identifiers.

```
StreamMessage smo = producer.session.createStreamMessage();
smo.writeBytes("123".getBytes(CCSID.getCodepage(((MQDestination) destination)
           .getIntProperty(WMQConstants.WMQ_CCSID))));
```

Obrázek 144. Psaní bajtů představujících řetězec v JMSStreamMessage s použitím cílové znakové sady

Příklady

Odesílání a příjem JMSTextmessage

Textová zpráva nemůže obsahovat text v různých znakových sadách. Příklad ukazuje text v různých znakových sadách, který je odeslán ve dvou různých zprávách.

```
TextMessage tmo = session.createTextMessage();
tmo.setText("Sent in the character set defined for the destination");
producer.send(tmo);
```

Obrázek 145. Odeslat textovou zprávu ve znakové sadě definované v místě určení

```
TextMessage tmo = session.createTextMessage();
tmo.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET, 37);
tmo.setText("Sent in EBCDIC character set 37");
producer.send(tmo);
```

Obrázek 146. Odeslat textovou zprávu v produktu ccsid 37

```
TextMessage tmi = (TextMessage)consumer.receive();
System.out.println(tmi.getText());
...
Sent in the character set defined for the destination
```

Obrázek 147. Přijmout textovou zprávu

Příklady kódování

Příklady, které ukazují číslo odesílané v kódování, definuje místo určení. Všimněte si, že je třeba nastavit vlastnost JMS_IBM_ENCODING hodnoty JMSBytesMessage na hodnotu zadanou pro místo určení.

```
StreamMessage smo = session.createStreamMessage();
smo.writeInt(256);
producer.send(smo);
...
StreamMessage smi = (StreamMessage)consumer.receive();
System.out.println(smi.readInt());
...
256
```

Obrázek 148. Odesílání čísla s použitím kódování cíle v produktu JMSStreamMessage

```
BytesMessage bmo = session.createBytesMessage();
bmo.writeInt(256);
int encoding = ((MQDestination) (destination)).getIntProperty
    (WMQConstants.WMQ_ENCODING)
bmo.setIntProperty(WMQConstants.JMS_IBM_ENCODING, encoding);
producer.send(bmo);
...
BytesMessage bmi = (BytesMessage)consumer.receive();
System.out.println(bmi.readInt());
...
256
```

Obrázek 149. Odesílání čísla s použitím kódování cíle v produktu JMSBytesMessage

Odesílání a příjem textu v JMSBytesMessage

Kód v produktu Obrázek 150 na stránce 818 odesílá řetězec do pole BytesMessage. Pro zjednodušení příklad odesílá jeden řetězec, pro který je vhodnější JMSTextMessage . Chcete-li přijmout textový řetězec v bajtové zprávě obsahující kombinaci typů, musíte znát délku řetězce v bajtech, kterému se říká *TEXT_LENGTH* v produktu Obrázek 151 na stránce 818. I v případě řetězce s pevným počtem znaků může být délka znázornění bajtů delší.

```
BytesMessage bytes = session.createBytesMessage();
String codePage = CCSID.getCodepage(((MQDestination) destination)
                  .getIntProperty(WMQConstants.WMQ_CCSID));
bytes.writeBytes("In the destination code page".getBytes(codePage));
producer.send(bytes);
```

Obrázek 150. Odeslání String v JMSBytesMessage

```
BytesMessage message = (BytesMessage)consumer.receive();
int TEXT_LENGTH = new Long(message.getBodyLength())).intValue();
byte[] textBytes = new byte[TEXT_LENGTH];
message.readBytes(textBytes, TEXT_LENGTH);
String codePage = message.getStringProperty(WMQConstants.JMS_IBM_CHARACTER_SET);
String textString = new String(textBytes, codePage);
```

Obrázek 151. Příjem String z JMSBytesMessage

Související pojmy

Přístupy k převodu zpráv JMS

Pro návrháře aplikací JMS je otevřeno řada přístupů pro převod dat. Tyto přístupy se nevylučují; některé aplikace budou pravděpodobně používat kombinaci těchto přístupů. Pokud si aplikace vyměňuje pouze text nebo si vyměňuje zprávy pouze s jinými aplikacemi JMS, obvykle nepovažujete převod dat za normální. Převod dat se provádí automaticky pro vás, v produktu WebSphere MQ.

Převod dat správce front

Převod dat správce front byl vždy k dispozici pro aplikace mimo platformu JMS, které přijímají zprávy od klientů JMS. Vzhledem k tomu, že klienti JMS přijímají zprávy od V7.0, používají také převod dat správce front. Počínaje verzí 7.0.1.5nebo 7.0.1.4 s opravou APAR IC72897je konverze dat správce front volitelná.

Související úlohy

Výměna formátovaného záznamu s aplikací jinou než JMS

Postupujte podle kroků doporučených v této úloze pro návrh a sestavení uživatelské procedury pro převod dat a aplikaci klienta JMS, která může vyměňovat zprávy s aplikací mimo platformu JMS pomocí produktu JMSBytesMessage. Výměnu formátované zprávy s jinou aplikací než JMS se může uskutečnit s voláním ukončení konverze dat nebo bez volání ukončení konverze dat.

Související odkazy

Typy a konverze zpráv rozhraní JMS
Výběr typu zprávy ovlivňuje váš přístup k převodu zpráv. Interakce zprávy o převodu zpráv a typu zprávy jsou popsány pro typy zpráv JMS, `JMSObjectMessage`, `JMSTextMessage`, `JMSMapMessage`, `JMSStreamMessage`a `JMSBytesMessage`.

Převod dat správce front
Převod dat správce front byl vždy k dispozici pro aplikace mimo platformu JMS, které přijímají zprávy od klientů JMS. Vzhledem k tomu, že klienti JMS přijímají zprávy od V7.0, používají také převod dat správce front. Počínaje verzí 7.0.1.5nebo 7.0.1.4 s opravou APAR IC72897je konverze dat správce front volitelná.

Správce front může převádět znaková data a číselná data v datech zprávy pomocí hodnot `CodedCharacterSetId`, `Encoding`a `Format` nastavených pro data zprávy. Pro aplikace mimo platformu JMS byla možnost převodu vždy k dispozici nastavením volby `GetMessageOption`, `GMO_CONVERT`. Schopnost převodu správce front není k dispozici pro aplikaci JMS přijímající zprávu do V7.0.

Převod správce front můžete použít dříve, než V7.0, s aplikací klienta JMS, která odešle zprávu. Klient služby JMS sestavuje formátovaný záznam, nastavuje atributy `CodedCharacterSetId`, `Encoding`a `Format` odpovídající datům umístěnými ve zprávě. Přijímací aplikace mimo rozhraní JMS přečte zprávu pomocí příkazu `GMO_CONVERT`a způsobí, že bude volána uživatelská procedura pro převod dat. Uživatelská procedura pro převod dat je sdílená knihovna, která má název nastavený v poli `Format` .

Od verze V7.0je správce front schopen převést zprávy, které jsou odesílány klientům JMS. Od verze 7.0.0.0 do 7.0.1.4 včetně je převod správce front vždy volán pro klienty JMS. Od verze 7.0.1.5nebo z verze 7.0.1.4 s použitím opravy APAR IC72897 je převod správce front řízen nastavením vlastnosti cíle `WMQ_RECEIVE_CONVERSION`na hodnotu `WMQ_RECEIVE_CONVERSION_QMGR`nebo `WMQ_RECEIVE_CONVERSION_CLIENT_MSG`. Výchozí nastavení je `WMQ_RECEIVE_CONVERSION_CLIENT_MSG` , které odpovídá chování produktu WebSphere MQ V6.0, které nepodporuje převod dat správce front pro klienty JMS. Aplikace může změnit nastavení místa určení:

```
((MQDestination)destination).setIntProperty(
              WMQConstants.WMQ_RECEIVE_CONVERSION,
              WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
```

Nebo,

```
((MQDestination)destination).setReceiveConversion
             (WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
```

Obrázek 152. Povolit převod dat správce front

Převod dat správce front pro klienta JMS se provádí, když klient volá metodu `consumer.receive` . Textová data jsou při výchozím nastavení transformována do formátu UTF-8 (1208). Následné čtení a získání textu dekóduje text v přijatých datech z UTF-8, čímž se vytvoří textová primitiva Java ve svém interním kódování Unicode. UTF-8 není jediná cílová znaková sada z převodu dat správce front. Nastavením vlastnosti cíle `WMQ_RECEIVE_CCSID` si můžete vybrat jiný CCSID.

Aplikace může také změnit nastavení cíle, například nastavení na 437, DOS-US:

```
((MQDestination)destination).setIntProperty
            (WMQConstants.WMQ_RECEIVE_CCSID, 437);
```

Nebo,

```
((MQDestination)destination).setReceiveCCSID(437);
```

Obrázek 153. Nastavit cílovou kódovou sadu znaků pro převod správce front

Důvod změny WMQ_RECEIVE_CCSID je specializovaný; zvolený CCSID nečiní žádný rozdíl pro textové objekty vytvořené v prostředí JVM. Některá prostředí JVM, na některých platformách, však nemusí být schopna zpracovat převod z CCSID textu ve zprávě do Unicode. Tato volba vám dává volbu CCSID pro jakýkoli text doručený klientovi ve zprávě. Některé klientské platformy JMS mají problémy s textem zprávy dodávanými v UTF-8.

Kód JMS je ekvivalentem tučného textu v kódu jazyka C v produktu Obrázek 154 na stránce 820,

```
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 154. Úsek kódu z amqsget0.c

Poznámka:

Převod správce front se provádí pouze na datech zprávy, která mají známý formát WebSphere MQ . MQSTR, nebo MQCIH jsou příklady známých formátů, které jsou předdefinované. Známý formát může být také uživatelem definovaný formát, pokud jste zadali uživatelskou proceduru pro převod dat.

Zprávy vytvořené jako JMSTextMessage, JMSMapMessage a JMSStreamMessage, mají formát MQSTR a mohou být převedeny správcem front.

Související pojmy

Přístupy k převodu zpráv JMS
Pro návrháře aplikací JMS je otevřeno řada přístupů pro převod dat. Tyto přístupy se nevylučují; některé aplikace budou pravděpodobně používat kombinaci těchto přístupů. Pokud si aplikace vyměňuje pouze text nebo si vyměňuje zprávy pouze s jinými aplikacemi JMS, obvykle nepovažujete převod dat za normální. Převod dat se provádí automaticky pro vás, v produktu WebSphere MQ.

Převod a kódování zpráv klienta JMS
Jsou vypsány metody, které se používají k provedení konverze a kódování zpráv klienta JMS, s příklady kódu jednotlivých typů konverze.

“Vyvolání uživatelské procedury pro převod dat” na stránce 400
Uživatelská procedura pro převod dat je uživatelská procedura, která přijímá řízení během zpracování volání MQGET.

Související úlohy

Výměna formátovaného záznamu s aplikací jinou než JMS
Postupujte podle kroků doporučených v této úloze pro návrh a sestavení uživatelské procedury pro převod dat a aplikaci klienta JMS, která může vyměňovat zprávy s aplikací mimo platformu JMS pomocí produktu `JMSBytesMessage`. Výměnu formátované zprávy s jinou aplikací než JMS se může uskutečnit s voláním ukončení konverze dat nebo bez volání ukončení konverze dat.

Související odkazy

Typy a konverze zpráv rozhraní JMS
Výběr typu zprávy ovlivňuje váš přístup k převodu zpráv. Interakce zprávy o převodu zpráv a typu zprávy jsou popsány pro typy zpráv JMS, `JMSObjectMessage`, `JMSTextMessage`, `JMSMapMessage`, `JMSStreamMessagea` `JMSBytesMessage`.

Výměna formátovaného záznamu s aplikací jinou než JMS
Postupujte podle kroků doporučených v této úloze pro návrh a sestavení uživatelské procedury pro převod dat a aplikaci klienta JMS, která může vyměňovat zprávy s aplikací mimo platformu JMS pomocí produktu `JMSBytesMessage`. Výměnu formátované zprávy s jinou aplikací než JMS se může uskutečnit s voláním ukončení konverze dat nebo bez volání ukončení konverze dat.

Než začnete

Je možné, že budete schopni navrhnout jednodušší řešení pro výměnu zpráv s aplikací mimo platformu JMS pomocí produktu `JMSTextMessage`. Před provedením kroků uvedených v této úloze tuto možnost eliminujte.

Informace o této úloze

Klient platformy JMS je snazší pro zápis, pokud se nezapojuje do podrobností formátování zpráv JMS vyměňovaných s jinými klienty JMS. Pokud se jedná o typ zprávy `JMSTextMessage`, `JMSMapMessage`, `JMSStreamMessage`nebo `JMSObjectMessage`, produkt WebSphere MQ se bude hledat po podrobnostech formátování zprávy. Produkt WebSphere MQ pracuje s rozdíly v kódových stránkách a číselném kódování na různých platformách.

Tyto typy zpráv můžete použít k výměně zpráv s aplikacemi jiného typu než JMS. Chcete-li tak učinit, musíte porozumět způsobu, jakým jsou tyto zprávy vytvářeny třídami WebSphere MQ pro platformu JMS. Je možné, že budete moci upravit aplikaci, která není typu JMS, aby interpretoval zprávy, viz “Mapování zpráv JMS na zprávy produktu WebSphere MQ” na stránce 784.

Výhoda použití jednoho z těchto typů zpráv znamená, že programování klienta JMS nezávisí na typu aplikace, se kterou si vyměňuje zprávy. Nevýhodou je, že by mohla vyžadovat úpravu jiného programu a možná nebudete moci změnit jiný program.

Alternativním přístupem je napsat klientskou aplikaci JMS, která dokáže pracovat s existujícími formáty zpráv. Často existující zprávy mají pevný formát a obsahují směsici neformátovaných dat, textu a čísel. Postupujte podle kroků uvedených v této úloze a příkladu klienta JMS v produktu “Psaní tříd pro zapouzdření rozvržení záznamu v produktu JMSBytesMessage” na stránce 824jako výchozí bod pro sestavení klienta JMS, který může vyměňovat formátované záznamy s aplikacemi mimo systém JMS.

Postup

1. Definujte rozvržení záznamu nebo použijte jednu z předdefinovaných tříd záhlaví produktu WebSphere MQ .

   Informace o zpracování předdefinovaných záhlaví produktu WebSphere MQ naleznete v tématu Práce se záhlavími zpráv produktu WebSphere MQ.

   Obrázek 155 na stránce 822 je příklad rozvržení záznamu pevné délky, které lze zpracovat obslužným programem pro převod dat.
2. Vytvořte uživatelskou proceduru pro převod dat.

Postupujte podle pokynů v části Psaní výstupního programu pro převod dat , abyste zapsali uživatelskou proceduru pro převod dat.

Chcete-li vyzkoušet příklad v souboru “Psaní tříd pro zapouzdření rozvržení záznamu v produktu JMSBytesMessage” na stránce 824, pojmenujte ukončení převodu dat MYRECORD.

3. Zapsat třídy Java pro zapouzdření rozvržení záznamu a odeslání a přijetí záznamu. Dva přístupy, které můžete provést, jsou:
   - Zapište třídu na čtení a zapíše JMSBytesMessage , která obsahuje záznam; viz “Psaní tříd pro zapouzdření rozvržení záznamu v produktu JMSBytesMessage” na stránce 824.
   - Zapište třídu rozšiřující com.ibm.mq.header.Header , abyste definovali datovou strukturu záznamu; viz Vytváření tříd pro nové typy záhlaví.
4. Rozhodněte, která kódovaná znaková sada se má vyměňovat ve zprávách.

   Viz “Výběr přístupu k převodu zpráv: receiver makes good” na stránce 803.
5. Nakonfigurujte cíl pro výměnu zpráv typu -type produktu MQbez záhlaví JMS MQRFH2 .

   Odesílající i přijímající místo určení musí být konfigurováno pro výměnu zpráv typu -type produktu MQ. Stejné místo určení můžete použít pro odeslání i příjem.

   Aplikace může přepsat vlastnost těla cílové zprávy:

   ```
   ((MQDestination)destination).setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_MQ);
   ```

   Příklad v produktu “Psaní tříd pro zapouzdření rozvržení záznamu v produktu JMSBytesMessage” na stránce 824 potlačí vlastnost těla zprávy cíle a bude odeslána zpráva se stylem MQ.
6. Otestujte řešení s aplikacemi JMS a jinými aplikacemi než JMS

   Užitečné nástroje pro testování ukončení konverze dat jsou:
   - Ukázkový program amqsgetc0.c je užitečný k testování příjmu zprávy odeslané klientem JMS. Prohlédněte si navrhované úpravy, abyste použili ukázkové záhlaví, RECORD.h, v produktu Obrázek 156 na stránce 823. S úpravami obdrží produkt amqsgetc0.c zprávu odeslanou příkladem klienta JMS TryMyRecord.java; viz “Psaní tříd pro zapouzdření rozvržení záznamu v produktu JMSBytesMessage” na stránce 824.
   - Ukázkový program pro procházení produktu WebSphere MQ amqsbcg0.cje užitečný ke kontrole obsahu záhlaví zprávy, záhlaví JMS, modulu MQRFH2a obsahu zprávy.
   - Program **rfhutil** , který byl dříve k dispozici v balíku SupportPac IH03, umožňuje zachycení a uložení testovacích zpráv v souborech a jejich použití k řízení toků zpráv. Výstupní zprávy lze také číst a zobrazovat v různých formátech. Formáty zahrnují dva typy XML, stejně jako porovnání oproti zakladači COBOL. Data mohou být ve formátu EBCDIC nebo ASCII. Do zprávy lze přidat záhlaví RFH2 , než bude odeslána zpráva.

   Pokusíte-li se o příjem zpráv pomocí upraveného ukázkového programu amqsgetc0.c a získání chyby s kódem příčiny 2080, zkontrolujte, zda má zpráva MQRFH2. Úpravy předpokládají, že zpráva byla odeslána do místa určení, které nespecifikuje žádné MQRFH2.

Příklady

```
struct RECORD { MQCHAR StrucID[4];
                MQLONG Version;
                MQLONG StructLength;
                MQLONG Encoding;
                MQLONG CodeCharSetId;
                MQCHAR Format[8];
                MQLONG Flags;
                MQCHAR RecordData[32];
    };
```

Obrázek 155. RECORD.h

- Deklarovat strukturu dat produktu RECORD.h

```
struct tagRECORD {
   MQCHAR4    StrucId;
   MQLONG     Version;
   MQLONG     StrucLength;
   MQLONG     Encoding;
   MQLONG     CCSID;
   MQCHAR8    Format;
   MQLONG     Flags;
   MQCHAR32   RecordData;
 };
typedef struct tagRECORD RECORD;
   typedef RECORD MQPOINTER PRECORD;
   RECORD record;
   PRECORD pRecord = &(record);
```

- Upravte volání MQGET tak, aby používal RECORD,

  1. Před úpravou:

```
    MQGET(Hcon,            /* connection handle     */
          Hobj,            /* object handle         */
          &md,             /* message descriptor    */
          &gmo,            /* get message options   */
          buflen,          /* buffer length         */
          buffer,          /* message buffer        */
          &messlen,        /* message length        */
          &CompCode,       /* completion code       */
          &Reason);        /* reason code           */
```

  2. Po úpravě:

```
    MQGET(Hcon,            /* connection handle     */
          Hobj,            /* object handle         */
          &md,             /* message descriptor    */
          &gmo,            /* get message options   */
          sizeof(RECORD),  /* buffer length         */
          pRecord,         /* message buffer        */
          &messlen,        /* message length        */
          &CompCode,       /* completion code       */
          &Reason);        /* reason code           */
```

- Změnit tiskový příkaz,

  1. Zdroj:

```
buffer[messlen] = '\0';            /* add terminator          */
printf("message <%s>\n", buffer);
```

  2. Do:

```
/* buffer[messlen] = '\0';            add terminator          */
printf("ccsid <%d>, flags <%d>, message <%32.32s>\n \0",
        md.CodedCharSetId, record.Flags, record.RecordData);
```

Obrázek 156. Upravit amqsget0.c

Související pojmy

Přístupy k převodu zpráv JMS

Pro návrháře aplikací JMS je otevřeno řada přístupů pro převod dat. Tyto přístupy se nevylučují; některé aplikace budou pravděpodobně používat kombinaci těchto přístupů. Pokud si aplikace vyměňuje pouze text nebo si vyměňuje zprávy pouze s jinými aplikacemi JMS, obvykle nepovažujete převod dat za normální. Převod dat se provádí automaticky pro vás, v produktu WebSphere MQ.

Převod a kódování zpráv klienta JMS

Jsou vypsány metody, které se používají k provedení konverze a kódování zpráv klienta JMS, s příklady kódu jednotlivých typů konverze.

Převod dat správce front

Převod dat správce front byl vždy k dispozici pro aplikace mimo platformu JMS, které přijímají zprávy od klientů JMS. Vzhledem k tomu, že klienti JMS přijímají zprávy od V7.0, používají také převod dat správce front. Počínaje verzí 7.0.1.5nebo 7.0.1.4 s opravou APAR IC72897je konverze dat správce front volitelná.

Související odkazy

Typy a konverze zpráv rozhraní JMS
Výběr typu zprávy ovlivňuje váš přístup k převodu zpráv. Interakce zprávy o převodu zpráv a typu zprávy jsou popsány pro typy zpráv JMS, `JMSObjectMessage`, `JMSTextMessage`, `JMSMapMessage`, `JMSStreamMessage`a `JMSBytesMessage`.

Obslužný program pro vytvoření kódu ukončení převodu

Psaní tříd pro zapouzdření rozvržení záznamu v produktu JMSBytesMessage
Účelem této úlohy je prozkoumat například způsob, jak kombinovat převod dat a pevné rozvržení záznamu v produktu `JMSBytesMessage`. V této úloze vytvoříte některé třídy Java pro výměnu vzorové struktury záznamů v produktu `JMSBytesMessage`. Můžete upravit příklad pro zápis tříd za účelem výměny jiných struktur záznamů.

`JMSBytesMessage` je nejlepší volbou typu zprávy JMS pro výměnu záznamů datového typu smíšených dat s jinými programy než s programy JMS. Neobsahuje žádná další data vložená do těla zprávy poskytovatelem JMS. Je proto nejlepším volbou typu zprávy, který má být použit, pokud klientský program JMS spolupracuje s existujícím programem IBM WebSphere MQ . Hlavní úkol při použití `JMSBytesMessage` je v souladu s kódováním a znakovou sadou očekávanou jiným programem. Řešením je vytvořit třídu, která bude zapouzdřit záznam. Třída, která zapouzdřuje čtení a zapisuje `JMSBytesMessage`pro určitý typ záznamu, usnadňuje odesílání a příjem záznamů v pevném formátu v programu JMS. Díky zachycení obecných aspektů rozhraní v abstraktní třídě může být velká část řešení znovu použita pro různé formáty záznamu. Ve třídách, které rozšiřují abstraktní generickou třídu, lze implementovat různé formáty záznamů.

Alternativním přístupem je rozšíření třídy `com.ibm.mq.headers.Header` . Třída `Header` má metody, jako např. `addMQLONG`, aby vytvářel formát záznamu v deklarativním způsobem. Nevýhodou použití třídy `Header` je získávání a nastavení atributů pomocí složitějšího interpretačního rozhraní. Oba přístupy vedou ke stejnému množství kódu aplikace.

`JMSBytesMessage` může zapouzdřit pouze jeden formát, kromě `MQRFH2`, v jedné zprávě, pokud každý záznam nepoužívá stejný formát, kódovanou znakovou sadu a kódování. Formát, kódování a znaková sada `JMSBytesMessage` jsou vlastnosti všech zpráv, které následují za serverem `MQRFH2`. Příklad je zapsán za předpokladu, že produkt `JMSBytesMessage` obsahuje pouze jeden záznam uživatele.

Než začnete

1. Vaše odbornost: Je třeba, abyste byli obeznámeni s programováním Java a s platformou JMS. Nejsou poskytnuty žádné pokyny o nastavení vývojového prostředí Java. Je výhodné, že jste napsali program pro výměnu `JMSTextMessage`, `JMSStreamMessage`nebo `JMSMapMessage`. Poté můžete zobrazit rozdíly ve výměně zpráv pomocí produktu `JMSBytesMessage`.
2. Příklad vyžaduje IBM WebSphere MQ V7.0.
3. Příklad byl vytvořen s použitím perspektivy Java pracovní plochy Eclipse . Vyžaduje prostředí JRE 6.0 nebo vyšší. K vývoji a spouštění tříd Java můžete použít perspektivu Java v produktu IBM WebSphere MQ Explorer. Můžete také použít své vlastní vývojové prostředí Java.
4. Použití Průzkumníka IBM WebSphere MQ provádí nastavení testovacího prostředí a ladění, je jednodušší než použití obslužných programů příkazového řádku.

Informace o této úloze

Jste vedeni vytvořením dvou tříd: `RECORD` a `MyRecord`. Společně tyto dvě třídy zapouzdřují záznam v pevném formátu. Mají metody pro získání a nastavení atributů. Metoda get čte záznam z `JMSBytesMessage` a metoda put zapíše záznam do `JMSBytesMessage`.

Účelem úlohy není vytvoření třídy jakosti výroby, kterou lze použít opakovaně. Můžete se rozhodnout použít příklady v úloze, abyste mohli začít pracovat na svých vlastních třídách. Účelem této úlohy

je poskytnout vám naváděcí poznámky, především o použití znakových sad, formátů a kódování při použití JMSBytesMessage. Je popsán každý krok při vytváření tříd a jsou popsány aspekty použití JMSBytesMessage, které se někdy přehlížejí.

Třída RECORD je abstraktní a definuje některá společná pole pro záznam uživatele. Společná pole jsou modelována na standardním rozvržení záhlaví IBM WebSphere MQ , které má zvýrazňovač, verzi a délku pole. Pole kódování, znaková sada a formát, která je nalezena v mnoha záhlavích IBM WebSphere MQ , jsou vynechána. Další záhlaví nemůže následovat uživatelem definovaný formát. Třída MyRecord , která rozšiřuje třídu RECORD tak, že doslova rozšiřuje záznam o další pole uživatele. Soubor JMSBytesMessagevytvořený třídami může být zpracován uživatelskou procedurou pro převod dat správce front.

Produkt “Třídy použité ke spuštění příkladu” na stránce 831 obsahuje úplný výpis položek RECORD a MyRecord. Obsahuje také výpisy dalších tříd "lešení" k testování produktů RECORD a MyRecord. Další třídy jsou:

TryMyRecord
Hlavní program pro testování RECORD a MyRecord.

EndPoint
Abstraktní třída, která zapouzdřuje připojení JMS, cíl a relaci v jedné třídě. Jeho rozhraní právě odpovídá potřebám testování tříd RECORD a MyRecord . Není zavedený vzor návrhu pro zápis aplikací JMS.

Poznámka: Třída Endpoint obsahuje tento řádek kódu po vytvoření místa určení:

```
((MQDestination)destination).setReceiveConversion
                            (WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
```

V produktu V7.0z verze V7.0.1.5je nutné zapnout převod správce front. Podle výchozího nastavení je tato volba zakázána. Ve verzi V7.0je standardně povolen převod správce front až na V7.0.1.4 a tento řádek kódu způsobuje chybu.

MyProducer a MyConsumer
Třídy, které rozšiřují prostor EndPointa vytvářejí MessageConsumer a MessageProducer, jsou připojeny a připraveny přijímat požadavky.

Všechny třídy dohromady tvoří úplnou aplikaci, se kterou můžete sestavit a experimentovat s cílem pochopit, jak používat převod dat v produktu JMSBytesMessage.

Postup

1. Vytvořte abstraktní třídu pro zapouzdření standardních polí v záhlaví IBM WebSphere MQ s výchozím konstruktorem. Později rozšíříte třídu tak, abyste přizpůsobili záhlaví vašim požadavkům.

```
public abstract class RECORD implements Serializable {
    private static final long serialVersionUID = -1616617232750561712L;
    protected final static int UTF8 = 1208;
    protected final static int MQLONG_LENGTH = 4;
    protected final static int RECORD_STRUCT_ID_LENGTH = 4;
    protected final static int RECORD_VERSION_1 = 1;
    protected final String RECORD_STRUCT_ID = "BLNK";
    protected final String RECORD_TYPE = "BLANK   ";
    private String structID = RECORD_STRUCT_ID;
    private int version = RECORD_VERSION_1;
    private int structLength = RECORD_STRUCT_ID_LENGTH + MQLONG_LENGTH * 2;
    private int headerEncoding = WMQConstants.WMQ_ENCODING_NATIVE;
    private String headerCharset = "UTF-8";
    private String headerFormat = RECORD_TYPE;

    public RECORD() {
        super();
    }
```

Poznámka:

a. Atributy, structID až nextFormat, jsou vypsány v pořadí, ve kterém jsou uvedeny ve standardním záhlaví zprávy IBM WebSphere 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. Java používá znakové sady a IBM WebSphere MQ zprávy používají identifikátory kódované znakové sady. Ukázkový kód používá znakové sady.

d. `int` je serializován do `MQLONG` pomocí IBM WebSphere MQ. `MQLONG` je 4 bajty.

2. Vytvoření metod getter a setter pro soukromé atributy.

a) Vytvořte nebo vygenerujte metody getter:

```
    public String getHeaderFormat() { return headerFormat; }
    public int getHeaderEncoding() { return headerEncoding;    }
    public String getMessageCharset() { return headerCharset; }
    public int getMessageEncoding() { return headerEncoding; }
    public String getStructID() { return structID; }
    public int getStructLength() { return structLength; }
    public int getVersion() { return version; }
```

b) Vytvořte nebo vygenerujte metody setter:

```
    public void setHeaderCharset(String charset) {
        this.headerCharset = charset; }
    public void setHeaderEncoding(int encoding) {
        this.headerEncoding = encoding; }
    public void setHeaderFormat(String headerFormat) {
        this.headerFormat = headerFormat; }
    public void setStructID(String structID) {
        this.structID = structID; }
    public void setStructLength(int structLength) {
        this.structLength = structLength; }
    public void setVersion(int version) {
        this.version = version; }
}
```

3. Vytvořte konstruktor pro vytvoření instance RECORD z `JMSBytesMessage`.

```
public RECORD(BytesMessage message) throws JMSException, IOException,
       MQDataException {
    super();
    setHeaderCharset(message.getStringProperty(WMQConstants.JMS_IBM_CHARACTER_SET));
    setHeaderEncoding(message.getIntProperty(WMQConstants.JMS_IBM_ENCODING));
    byte[] structID = new byte[RECORD_STRUCT_ID_LENGTH];
    message.readBytes(structID, RECORD_STRUCT_ID_LENGTH);
    setStructID(new String(structID, getMessageCharset()));
    setVersion(message.readInt());
    setStructLength(message.readInt());
}
```

Poznámka:

a. Hodnoty `messageCharset` a `messageEncoding`jsou zachyceny ve vlastnostech zprávy, protože přepisují hodnoty nastavené pro místo určení. `format` není aktualizován. Příklad nekontroluje chybu. Je-li volán konstruktor `Record(BytesMessage)` , předpokládá se, že `JMSBytesMessage` je typ zprávy RECORD . Řádek "`setStructID(new String(structID, getMessageCharset()))`" nastaví zvýrazňovač očí.

b. Čáry kódu, které dokončují deserializační pole metody ve zprávě, v pořadí aktualizující výchozí hodnoty nastavené v instanci RECORD.

4. Vytvořte metodu put pro zápis polí záhlaví do `JMSBytesMessage`.

```
protected BytesMessage put(MyProducer myProducer) throws IOException,
        JMSException, UnsupportedEncodingException {
    setHeaderEncoding(myProducer.getEncoding());
    setHeaderCharset(myProducer.getCharset());
    myProducer.setMQClient(true);
    BytesMessage bytes = myProducer.session.createBytesMessage();
    bytes.setStringProperty(WMQConstants.JMS_IBM_FORMAT, getHeaderFormat());
    bytes.setIntProperty(WMQConstants.JMS_IBM_ENCODING, getHeaderEncoding());
    bytes.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET,
                         myProducer.getCCSID());
    bytes.writeBytes(String.format("%1$-" + RECORD_STRUCT_ID_LENGTH + "."
```

```
                        + RECORD_STRUCT_ID_LENGTH + "s", getStructID())
                        .getBytes(getMessageCharset()), 0, RECORD_STRUCT_ID_LENGTH);
        bytes.writeInt(getVersion());
        bytes.writeInt(getStructLength());
        return bytes;
    }
```

Poznámka:

a. MyProducer zapouzdřuje JMS Connection, Destination, Sessiona MessageProducer v jedné třídě. MyConsumer, použije se později, zapouzdří JMS Connection, Destination, Sessiona MessageConsumer v jedné třídě.

b. Pokud je pro systém JMSBytesMessagekódování jiné než Native, musí být kódování nastaveno ve zprávě. Kódování cíle se zkopíruje do atributu kódování zpráv JMS_IBM_CHARACTER_SETa uloží se jako atribut třídy RECORD .

   i) "setMessageEncoding(myProducer.getEncoding());" volá příkaz "(((MQDestination) destination).getIntProperty(WMQConstants.WMQ_ENCODING));" k získání cílového kódování.

   ii) Produkt "Bytes.setIntProperty(WMQConstants.JMS_IBM_ENCODING, getMessageEncoding());" nastavuje kódování zpráv.

c. Znaková sada použitá k transformaci textu na bajty se získá z cíle a uloží se jako atribut třídy RECORD . Není nastavena ve zprávě, protože ji nepoužívá třídy IBM WebSphere MQ pro JMS při zápisu JMSBytesMessage.

   Volání "messageCharset = myProducer.getCharset();"

```
    public String getCharset() throws UnsupportedEncodingException,
            JMSException {
        return CCSID.getCodepage(getCCSID());
    }
```

   Z identifikátoru kódované znakové sady získá znakovou sadu jazyka Java.

   "CCSID.getCodepage(ccsid)" je v balíku com.ibm.mq.headers. Produkt ccsid se získává z jiné metody v produktu MyProducer, která se dotazuje na místo určení:

```
    public int getCCSID() throws JMSException {
        return (((MQDestination) destination)
                .getIntProperty(WMQConstants.WMQ_CCSID));
    }
```

d. Volba "myProducer.setMQClient(true);" přepíše cílové nastavení pro typ klienta a vynucuje jej klientovi IBM WebSphere MQ MQI. Možná dáte přednost vynechání této řádky kódu, protože zamlžuje administrativní chybu konfigurace.

   Volání "myProducer.setMQClient(true);":

```
((MQDestination) destination).setTargetClient(WMQConstants.WMQ_TARGET_DEST_MQ); }
if (!getMQDest()) setMQBody();
```

   Kód má vedlejší účinek nastavení stylu textu IBM WebSphere MQ na nespecifikovaný, pokud musí přepsat nastavení platformy JMS.

   Poznámka:

   Třídy IBM WebSphere MQ pro JMS zapisují identifikátor formátu, kódování a identifikátor znakové sady zprávy do deskriptoru zpráv, MQMDnebo do záhlaví JMS MQRFH2. Závisí na tom, zda má zpráva tělo stylu IBM WebSphere MQ . Nenastavujte pole MQMD ručně.

Existuje metoda pro ruční nastavení vlastností deskriptoru zpráv. Používá vlastnosti produktu JMS_IBM_MQMD_* . Chcete-li nastavit vlastnosti produktu JMS_IBM_MQMD_* , musíte nastavit vlastnost cíle WMQ_MQMD_WRITE_ENABLED :

```
((MQDestination)destination).setMQMDWriteEnabled(true);
```

Chcete-li si přečíst vlastnosti, musíte nastavit cílovou vlastnost WMQ_MQMD_READ_ENABLED.

JMS_IBM_MQMD_* použijte pouze v případě, že plně ovládíte celý informační obsah zprávy. Na rozdíl od vlastností produktu JMS_IBM_* neurčují vlastnosti produktu JMS_IBM_MQMD_* způsob, jakým třídy IBM WebSphere MQ pro platformu JMS vytváří zprávu JMS. Vlastnosti deskriptoru zpráv, které jsou v konfliktu s vlastnostmi zprávy JMS, je možné vytvořit.

e. Čáry kódu, které dokončí metodu, budou serializovat atributy ve třídě jako pole ve zprávě.

Atributy řetězce jsou doplněny mezerami. Řetězce jsou převedeny na bajty pomocí znakové sady definované pro záznam a oříznuty na délku polí zprávy.

5. Dokončete třídu přidáním importů.

```
package com.ibm.mq.id;
import java.io.IOException;
import java.io.Serializable;
import java.io.UnsupportedEncodingException;
import javax.jms.BytesMessage;
import javax.jms.JMSException;
import com.ibm.mq.constants.MQConstants;
import com.ibm.mq.headers.MQDataException;
import com.ibm.msg.client.wmq.WMQConstants;
```

6. Vytvořte třídu, abyste rozšířily třídu RECORD tak, aby zahrnovala další pole. Zahrnout výchozí konstruktor.

```
public class MyRecord extends RECORD {
    private static final long serialVersionUID = -370551723162299429L;
    private final static int FLAGS = 1;
    private final static String STRUCT_ID = "MYRD";
    private final static int DATA_LENGTH = 32;
    private final static String FORMAT = "MYRECORD";
    private int flags = FLAGS;
    private String recordData = "ABCDEFGHIJKLMNOPQRSTUVWXYZ012345";

    public MyRecord() {
        super();
        super.setStructID(STRUCT_ID);
        super.setHeaderFormat(FORMAT);
        super.setStructLength(super.getStructLength() + MQLONG_LENGTH
                + DATA_LENGTH);
    }
```

Poznámka:

a. Podtřída RECORD , MyRecord, upravuje velikost záhlaví, formát a délku záhlaví.

7. Vytvořte nebo vygenerujte metody getter a setter.

a) Vytvořte metody getter:

```
    public int getFlags() { return flags; }
    public String getRecordData() { return recordData; }Â .
```

b) Vytvořte metody Setter:

```
    public void setFlags(int flags) {
        this.flags = flags; }
    public void setRecordData(String recordData) {
        this.recordData = recordData; }
}
```

8. Vytvořte konstruktor pro vytvoření instance MyRecord z JMSBytesMessage.

```
    public MyRecord(BytesMessage message) throws JMSException, IOException,
            MQDataException {
```

```
        super(message);
        setFlags(message.readInt());
        byte[] recordData = new byte[DATA_LENGTH];
        message.readBytes(recordData, DATA_LENGTH);
        setRecordData(new String(recordData, super.getMessageCharset()));
    }
```

Poznámka:

a. Pole, která tvoří standardní šablonu zprávy, jsou přečteny nejprve třídou RECORD .

b. Text recordData se převede na String s použitím vlastnosti znakové sady pro zprávu.

9. Vytvořte statickou metodu pro získání zprávy od spotřebitele a vytvořte novou instanci produktu MyRecord .

```
    public static MyRecord get(MyConsumer myConsumer) throws JMSException,
            MQDataException, IOException {
        BytesMessage message = (BytesMessage) myConsumer.receive();
        return new MyRecord(message);
    }
```

Poznámka:

a. V příkladu z důvodu stručnosti je konstruktor MyRecord(BytesMessage) volán ze statické metody get. Obvykle se můžete oddělit od přijetí zprávy od vytvoření nové instance MyRecord .

10. Vytvořte metodu put k připojení polí zákazníků k JMSBytesMessage obsahujícímu záhlaví zprávy.

```
    public BytesMessage put(MyProducer myProducer) throws JMSException,
            IOException {
        BytesMessage bytes = super.put(myProducer);
        bytes.writeInt(getFlags());
        bytes.writeBytes(String.format("%1$-" + DATA_LENGTH + "."
                        + DATA_LENGTH + "s",getRecordData())
                        .getBytes(super.getMessageCharset()), 0, DATA_LENGTH);
        myProducer.send(bytes);
        return bytes;
    }
```

Poznámka:

a. Metoda volá v kódu serializovat atributy ve třídě MyRecord jako pole ve zprávě.

- Atribut recordData String je doplněn mezerami, převedený na bajty pomocí znakové sady definované pro záznam a zkrácen na délku pole RecordData .

11. Dokončete třídu přidáním příkazů include.

```
package com.ibm.mq.id;
import java.io.IOException;
import javax.jms.BytesMessage;
import javax.jms.JMSException;
import com.ibm.mq.headers.MQDataException;
```

Výsledky

Výsledky:

- Výsledky ze spuštění třídy TryMyRecord :
  - Odesílá se zpráva v kódované znakové sadě 37 a při použití uživatelské procedury pro převod správce front:

```
Out flags 1 text ABCDEFGHIJKLMNOPQRSTUVWXYZ012345 Encoding 546 CCSID 37 MQ true
Out flags 1 text ABCDEFGHIJKLMNOPQRSTUVWXYZ012345 Encoding 546 CCSID 37 MQ true
In  flags 1 text ABCDEFGHIJKLMNOPQRSTUVWXYZ012345 Encoding 273 CCSID UTF-8
```

– Odesílá se zpráva v kódované znakové sadě 37 a *ne* pomocí 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 místo toho přijímá ji s použitím upraveného vzorku `amqsget0.c` . Upravená ukázka přijímá formátovaný záznam; viz Obrázek 156 na stránce 823 v “Výměna formátovaného záznamu s aplikací jinou než JMS” na stránce 821.
  – Odesílá se zpráva v kódované znakové sadě 37 a při použití uživatelské procedury pro převod správce front:

```
Sample AMQSGET0 start
ccsid <850>, flags <1>, message <ABCDEFGHIJKLMNOPQRSTUVWXYZ012345>
no more messages
Sample AMQSGET0 end
```

  – Odesílá se zpráva v kódované znakové sadě 37 a *ne* pomocí 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 s uživatelskou procedurou pro převod dat. Vytvořte třídy Java, nakonfigurujte IBM WebSphere MQa spusťte hlavní program, `TryMyRecord`; viz Obrázek 157 na stránce 831.

1. Nakonfigurujte produkt IBM WebSphere MQ a službu JMS pro spuštění příkladu. Pokyny jsou určeny ke spuštění příkladu na systému Windows.

   1. Vytvoření správce front

   ```
   crtmqm -sa -u SYSTEM.DEAD.LETTER.QUEUE QM1
   strmqm QM1
   ```

   2. Vytvoření fronty

   ```
   echo DEFINE QL('Q1') REPLACE | runmqsc QM1
   ```

   3. Vytvoření adresáře JNDI

   ```
   cd c:\
   md JNDI-Directory
   ```

   4. Přepněte na adresář bin platformy JMS
   Z tohoto místa musí být spuštěn program administrace platformy JMS. Cesta je *MQ_INSTALLATION_PATH*`\java\bin`.

   5. Vytvořte následující definice platformy 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
   ```

   6. Spusťte program JMSAdmin pro vytvoření prostředků JMS

   ```
   JMSAdmin < JMSQM1Q1.txt
   ```

2. Definice, které jste vytvořili pomocí Průzkumníka IBM WebSphere MQ , můžete vytvářet, měnit a procházet.
3. Spusťte příkaz `TryMyRecord`.

Třídy použité ke spuštění příkladu

Třídy uvedené v obrázcích Obrázek 157 na stránce 831 až Obrázek 162 na stránce 835 jsou k dispozici také v komprimovaném souboru; stáhněte soubor jm25529_.zip nebo jm25529_.tar.gz.

```
package com.ibm.mq.id;
public class TryMyRecord {
    public static void main(String[] args) throws Exception {
        MyProducer producer = new MyProducer();
        MyRecord outrec = new MyRecord();
        System.out.println("Out flags " + outrec.getFlags() + " text "
                + outrec.getRecordData() + " Encoding "
                + producer.getEncoding() + " CCSID " + producer.getCCSID()
                + " MQ " + producer.getMQDest());
        outrec.put(producer);
        System.out.println("Out flags " + outrec.getFlags() + " text "
                + outrec.getRecordData() + " Encoding "
                + producer.getEncoding() + " CCSID " + producer.getCCSID()
                + " MQ " + producer.getMQDest());
        MyRecord inrec = MyRecord.get(new MyConsumer());
        System.out.println("In  flags " + inrec.getFlags() + " text "
                + inrec.getRecordData() + " Encoding "
                + inrec.getMessageEncoding() + " CCSID "
                + inrec.getMessageCharset());
    }
}
```

Obrázek 157. TryMyRecord

```
package com.ibm.mq.id;
import java.io.IOException;
import java.io.Serializable;
import java.io.UnsupportedEncodingException;
import javax.jms.BytesMessage;
import javax.jms.JMSException;
import com.ibm.mq.constants.MQConstants;
import com.ibm.mq.headers.MQDataException;
import com.ibm.msg.client.wmq.WMQConstants;

public abstract class RECORD implements Serializable {
    private static final long serialVersionUID = -1616617232750561712L;
    protected final static int UTF8 = 1208;
    protected final static int MQLONG_LENGTH = 4;
    protected final static int RECORD_STRUCT_ID_LENGTH = 4;
    protected final static int RECORD_VERSION_1 = 1;
    protected final String RECORD_STRUCT_ID = "BLNK";
    protected final String RECORD_TYPE = "BLANK   ";
    private String structID = RECORD_STRUCT_ID;
    private int version = RECORD_VERSION_1;
    private int structLength = RECORD_STRUCT_ID_LENGTH + MQLONG_LENGTH * 2;
    private int headerEncoding = WMQConstants.WMQ_ENCODING_NATIVE;
    private String headerCharset = "UTF-8";
    private String headerFormat = RECORD_TYPE;

    public RECORD() {
        super();
    }

    public RECORD(BytesMessage message) throws JMSException, IOException,
            MQDataException {
        super();
        setHeaderCharset(message.getStringProperty(WMQConstants.JMS_IBM_CHARACTER_SET));
        setHeaderEncoding(message.getIntProperty(WMQConstants.JMS_IBM_ENCODING));
        byte[] structID = new byte[RECORD_STRUCT_ID_LENGTH];
        message.readBytes(structID, RECORD_STRUCT_ID_LENGTH);
        setStructID(new String(structID, getMessageCharset()));
        setVersion(message.readInt());
        setStructLength(message.readInt());
    }

    public String getHeaderFormat() { return headerFormat; }
    public int getHeaderEncoding() { return headerEncoding;    }
    public String getMessageCharset() { return headerCharset; }
    public int getMessageEncoding() { return headerEncoding; }
    public String getStructID() { return structID; }
    public int getStructLength() { return structLength; }
    public int getVersion() { return version; }


    protected BytesMessage put(MyProducer myProducer) throws IOException,
            JMSException, UnsupportedEncodingException {
        setHeaderEncoding(myProducer.getEncoding());
        setHeaderCharset(myProducer.getCharset());
        myProducer.setMQClient(true);
        BytesMessage bytes = myProducer.session.createBytesMessage();
        bytes.setStringProperty(WMQConstants.JMS_IBM_FORMAT, getHeaderFormat());
        bytes.setIntProperty(WMQConstants.JMS_IBM_ENCODING, getHeaderEncoding());
        bytes.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET,
                             myProducer.getCCSID());
        bytes.writeBytes(String.format("%1$-" + RECORD_STRUCT_ID_LENGTH + "."
                         + RECORD_STRUCT_ID_LENGTH + "s", getStructID())
                         .getBytes(getMessageCharset()), 0, RECORD_STRUCT_ID_LENGTH);
        bytes.writeInt(getVersion());
        bytes.writeInt(getStructLength());
        return bytes;
    }

    public void setHeaderCharset(String charset) {
        this.headerCharset = charset; }
    public void setHeaderEncoding(int encoding) {
        this.headerEncoding = encoding; }
    public void setHeaderFormat(String headerFormat) {
        this.headerFormat = headerFormat; }
    public void setStructID(String structID) {
        this.structID = structID; }
    public void setStructLength(int structLength) {
        this.structLength = structLength; }
    public void setVersion(int version) {
        this.version = version; }
}
```

Obrázek 158. RECORD

```
package com.ibm.mq.id;
import java.io.IOException;
import javax.jms.BytesMessage;
import javax.jms.JMSException;
import com.ibm.mq.headers.MQDataException;

public class MyRecord extends RECORD {
    private static final long serialVersionUID = -370551723162299429L;
    private final static int FLAGS = 1;
    private final static String STRUCT_ID = "MYRD";
    private final static int DATA_LENGTH = 32;
    private final static String FORMAT = "MYRECORD";
    private int flags = FLAGS;
    private String recordData = "ABCDEFGHIJKLMNOPQRSTUVWXYZ012345";

    public MyRecord() {
        super();
        super.setStructID(STRUCT_ID);
        super.setHeaderFormat(FORMAT);
        super.setStructLength(super.getStructLength() + MQLONG_LENGTH
                + DATA_LENGTH);
    }

    public MyRecord(BytesMessage message) throws JMSException, IOException,
            MQDataException {
        super(message);
        setFlags(message.readInt());
        byte[] recordData = new byte[DATA_LENGTH];
        message.readBytes(recordData, DATA_LENGTH);
        setRecordData(new String(recordData, super.getMessageCharset()));
    }

    public static MyRecord get(MyConsumer myConsumer) throws JMSException,
            MQDataException, IOException {
        BytesMessage message = (BytesMessage) myConsumer.receive();
        return new MyRecord(message);
    }

    public int getFlags() { return flags; }
    public String getRecordData() { return recordData; }Â .


    public BytesMessage put(MyProducer myProducer) throws JMSException,
            IOException {
        BytesMessage bytes = super.put(myProducer);
        bytes.writeInt(getFlags());
        bytes.writeBytes(String.format("%1$-" + DATA_LENGTH + "."
                        + DATA_LENGTH + "s",getRecordData())
                        .getBytes(super.getMessageCharset()), 0, DATA_LENGTH);
        myProducer.send(bytes);
        return bytes;
    }

    public void setFlags(int flags) {
        this.flags = flags; }
    public void setRecordData(String recordData) {
        this.recordData = recordData; }
}
```

Obrázek 159. MyRecord

```
package com.ibm.mq.id;
import java.io.UnsupportedEncodingException;
import javax.jms.Connection;
import javax.jms.ConnectionFactory;
import javax.jms.Destination;
import javax.jms.JMSException;
import javax.jms.Session;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import com.ibm.mq.headers.CCSID;
import com.ibm.mq.jms.MQDestination;
import com.ibm.msg.client.wmq.WMQConstants;
public abstract class EndPoint {
    public Context ctx;
    public ConnectionFactory cf;
    public Connection connection;
    public Destination destination;
    public Session session;
    protected EndPoint() throws NamingException, JMSException {
        System.setProperty("java.naming.provider.url", "file:/C:/JNDI-Directory");
        System.setProperty("java.naming.factory.initial",
                           "com.sun.jndi.fscontext.RefFSContextFactory");
        ctx = new InitialContext();
        cf = (ConnectionFactory) ctx.lookup("QM1");
        connection = cf.createConnection();
        destination = (Destination) ctx.lookup("Q1");
        ((MQDestination)destination).setReceiveConversion
                                    (WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
        session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE); }
    protected EndPoint(String cFactory, String dest) throws NamingException,
            JMSException {
        System.setProperty("java.naming.provider.url", "file:/C:/JNDI-Directory");
        System.setProperty("java.naming.factory.initial",
                           "com.sun.jndi.fscontext.RefFSContextFactory");
        ctx = new InitialContext();
        cf = (ConnectionFactory) ctx.lookup(cFactory);
        connection = cf.createConnection();
        destination = (Destination) ctx.lookup(dest);
        ((MQDestination)destination).setReceiveConversion
                                    (WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
        session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);  }
    public int getCCSID() throws JMSException {
        return (((MQDestination) destination)
                .getIntProperty(WMQConstants.WMQ_CCSID));  }
    public String getCharset() throws UnsupportedEncodingException,
            JMSException {
        return CCSID.getCodepage(getCCSID());  }
    public int getEncoding() throws JMSException {
        return (((MQDestination) destination)
                .getIntProperty(WMQConstants.WMQ_ENCODING));  }
    public boolean getMQDest() throws JMSException {
        if ((((MQDestination) destination).getMessageBodyStyle()
               == WMQConstants.WMQ_MESSAGE_BODY_MQ)
            || (((MQDestination) destination).getMessageBodyStyle()
                  == WMQConstants.WMQ_MESSAGE_BODY_UNSPECIFIED)
               && (((MQDestination) destination).getTargetClient()
                   == WMQConstants.WMQ_TARGET_DEST_MQ)))
            return true;
        else
            return false; }
    public void setCCSID(int ccsid) throws JMSException {
        ((MQDestination) destination).setIntProperty(WMQConstants.WMQ_CCSID,
                ccsid); }
    public void setEncoding(int encoding) throws JMSException {
        ((MQDestination) destination).setIntProperty(WMQConstants.WMQ_ENCODING,
                encoding); }
    public void setMQBody() throws JMSException {
        ((MQDestination) destination)
                    .setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_UNSPECIFIED);  }
    public void setMQBody(boolean mqbody) throws JMSException {
        if (mqbody)  ((MQDestination) destination)
                    .setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_MQ);
        else         ((MQDestination) destination)
                    .setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_JMS);  }
    public void setMQClient(boolean mqclient) throws JMSException {
       if (mqclient){
            ((MQDestination) destination).setTargetClient(WMQConstants.WMQ_TARGET_DEST_MQ);
            if (!getMQDest()) setMQBody();
       }
       else
            ((MQDestination) destination).setTargetClient(WMQConstants.WMQ_TARGET_DEST_JMS); }
    }
```

Obrázek 160. EndPoint

```
package com.ibm.mq.id;
import javax.jms.JMSException;
import javax.jms.Message;
import javax.jms.MessageProducer;
import javax.naming.NamingException;
public class MyProducer extends EndPoint {
    public MessageProducer producer;
    public MyProducer() throws NamingException, JMSException {
        super();
        producer = session.createProducer(destination); }
    public MyProducer(String cFactory, String dest) throws NamingException,
            JMSException {
        super(cFactory, dest);
        producer = session.createProducer(destination); }
    public void send(Message message) throws JMSException {
        producer.send(message); }
}
```

Obrázek 161. MyProducer

```
package com.ibm.mq.id;
import javax.jms.JMSException;
import javax.jms.Message;
import javax.jms.MessageConsumer;
import javax.naming.NamingException;
public class MyConsumer extends EndPoint {
    public MessageConsumer consumer;
    public MyConsumer() throws NamingException, JMSException {
        super();
        consumer = session.createConsumer(destination);
        connection.start();  }
    public MyConsumer(String cFactory, String dest) throws NamingException,
            JMSException {
        super(cFactory, dest);
        consumer = session.createConsumer(destination);
        connection.start();  }
    public Message receive() throws JMSException {
        return consumer.receive();  }
}
```

Obrázek 162. MyConsumer

Vytvoření a konfigurace továren připojení a cílů v třídách WebSphere MQ pro aplikaci JMS

Třídy produktu WebSphere MQ pro aplikaci JMS mohou vytvářet továrny připojení a cíle jejich načtením jako administrovaných objektů z oboru názvů rozhraní JNDI (Java Naming and Directory Interface) pomocí rozšíření IBM JMS nebo pomocí rozšíření WebSphere MQ JMS. Aplikace může také použít rozšíření JMS IBM nebo rozšíření produktu WebSphere MQ JMS k nastavení vlastností továren připojení a cílů.

Továrny připojení a cíle jsou počáteční body v toku logiky aplikace JMS. Aplikace používá objekt ConnectionFactory k vytvoření připojení k serveru systému zpráv a používá objekt Fronta nebo Téma jako cíl pro odesílání zpráv do nebo ze zdroje, ze kterého mají být přijímány zprávy. Aplikace proto musí vytvořit alespoň jednu továrnu připojení a jedno nebo více míst určení. Po vytvoření faktorie připojení nebo místa určení pak může aplikace nakonfigurovat objekt tak, že nastaví jednu nebo více jejích vlastností.

Stručně řečeno, aplikace může vytvářet a konfigurovat továrny připojení a cíle následujícími způsoby:

Použití JNDI k načtení spravovaných objektů
Administrátor může pomocí nástroje pro administraci produktu WebSphere MQ JMS nebo produktu WebSphere MQ Explorer vytvářet a konfigurovat továrny připojení a cíle jako spravované objekty v oboru názvů JNDI. Aplikace potom může načíst spravované objekty z oboru názvů JNDI. Po načtení spravovaného objektu může aplikace v případě potřeby nastavit nebo změnit jednu či více jejích vlastností buď pomocí rozšíření IBM JMS, nebo pomocí rozšíření WebSphere MQ JMS.

Použití rozšíření IBM JMS
Aplikace může používat rozšíření JMS společnosti IBM k dynamickému vytváření továren a cílů připojení za běhu. Aplikace nejprve vytvoří objekt továrny JmsFactorya poté použije metody tohoto objektu k vytvoření továren připojení a cílů. Po vytvoření faktorie připojení nebo místa určení může aplikace využívat metody zděděné z rozhraní kontextu JmsPropertyk nastavení vlastností.

Alternativně může aplikace použít identifikátor URI (Uniform Resource Identifier) k určení jedné nebo více vlastností cíle při vytváření místa určení.

Použití rozšíření WebSphere MQ JMS
Aplikace může také používat rozšíření WebSphere MQ JMS k dynamickému vytváření továren připojení a cílů v běhovém prostředí. Aplikace používá dodané konstruktory k vytvoření továren připojení a cílů. Po vytvoření faktorie připojení nebo místa určení může aplikace použít metody objektu k nastavení vlastností objektu. Aplikace může alternativně použít identifikátor URI k určení jedné nebo více vlastností cíle při vytváření místa určení.

Použití JNDI k načtení spravovaných objektů v aplikaci JMS

Chcete-li načíst spravované objekty z oboru názvů rozhraní JNDI (Java Naming and Directory Interface), musí aplikace platformy JMS vytvořit počáteční kontext a potom použít metodu lookup () k načtení objektů.

Než bude moci aplikace načítat spravované objekty z oboru názvů JNDI, musí nejprve vytvořit spravované objekty administrátor. Administrátor může pomocí nástroje pro administraci produktu WebSphere MQ JMS nebo produktu WebSphere MQ Explorer vytvářet a spravovat spravované objekty v oboru názvů rozhraní JNDI. Informace o způsobu použití nástroje pro administraci produktu WebSphere MQ JMS naleznete v příručce “Použití nástroje pro administraci produktu WebSphere MQ JMS” na stránce 902. Informace o tom, jak používat produkt WebSphere MQ Explorer, naleznete v nápovědě k produktu WebSphere MQ Explorer. Aplikační server však obvykle poskytuje vlastní úložiště pro spravované objekty a své vlastní nástroje pro vytváření a údržbu objektů.

Chcete-li načíst spravované objekty z oboru názvů JNDI, aplikace musí nejprve vytvořit počáteční kontext, jak je uvedeno v následujícím příkladu:

```
import javax.jms.*;
import javax.naming.*;
import javax.naming.directory.*;
.
.
.
String url = "ldap://server.company.com/o=company_us,c=us";
String icf = "com.sun.jndi.ldap.LdapCtxFactory";
.
java.util.Hashtable environment = new java.util.Hashtable();
environment.put(Context.PROVIDER_URL, url);
environment.put(Context.INITIAL_CONTEXT_FACTORY, icf);
Context ctx = new InitialDirContext(environment);
```

V tomto kódu mají řetězcové proměnné `url` a `icf` následující význam:

url
Uniform resource locator (URL) adresářové služby. Adresa URL může mít jeden z následujících formátů:

- `ldap://`*`hostname`*`/`*`contextName`*, pro adresářovou službu založenou na serveru LDAP
- `file:/`*`directoryPath`*, pro adresářovou službu založenou na lokálním systému souborů

if
Název třídy počáteční továrny kontextu, který může mít jednu z následujících hodnot:

- `com.sun.jndi.ldap.LdapCtxFactory`, pro adresářovou službu založenou na serveru LDAP
- `com.sun.jndi.fscontext.RefFSContextFactory`, pro adresářovou službu založenou na lokálním systému souborů

Všimněte si, že některé kombinace balíku JNDI a poskytovatele služeb LDAP (Lightweight Directory Access Protocol) mohou způsobit výskyt chyby LDAP 84. Chcete-li tento problém vyřešit, vložte před volání do kontextu InitialDirContext () následující řádek kódu:

```
environment.put(Context.REFERRAL, "throw");
```

Po získání počátečního kontextu může aplikace načíst administrované objekty z oboru názvů JNDI pomocí metody lookup (), jak je uvedeno v následujícím příkladu:

```
ConnectionFactory factory;
Queue queue;
Topic topic;
.
.
.
factory = (ConnectionFactory)ctx.lookup("cn=myCF");
queue = (Queue)ctx.lookup("cn=myQ");
topic = (Topic)ctx.lookup("cn=myT");
```

Tento kód načítá následující objekty z oboru názvů založené na protokolu LDAP:

- Objekt ConnectionFactory je svázán s názvem myCF .
- Objekt fronty svázaný s názvem myQ .
- Objekt tématu vázaný s názvem myT

Použití rozšíření IBM JMS

Třídy WebSphere MQ pro službu JMS obsahují sadu rozšíření rozhraní JMS API s názvem IBM JMS. Aplikace může tato rozšíření použít k dynamickému vytváření továren připojení a míst určení za běhu a k nastavení vlastností tříd produktu WebSphere MQ pro objekty JMS. Rozšíření lze použít s libovolným poskytovatelem systému zpráv.

Rozšíření IBM JMS jsou sadou rozhraní a tříd v následujících balících:

- com.ibm.msg.client.jms
- com.ibm.msg.client.services

Balíky lze nalézt v `com.ibm.mqjms.jar` , který se nachází v `<MQ_Install_Dir>/java/lib`.

Tato rozšíření poskytují následující funkce:

- Mechanizmus dynamického vytváření továren a cílů připojení za běhu v továrně namísto jejich načtení jako administrovaných objektů z oboru názvů rozhraní JNDI (Java Naming and Directory Interface).
- Sada metod pro nastavení vlastností tříd produktu WebSphere MQ pro objekty platformy JMS
- Sada tříd výjimek s metodami pro získání podrobných informací o problému
- Sada metod pro řízení trasování
- Sada metod pro získání informací o verzi pro třídy platformy WebSphere MQ pro JMS

Pokud se týká vytváření továren připojení a cílů dynamicky za běhu a nastavení a získávání jejich vlastností, rozšíření služby JMS IBM poskytují alternativní sadu rozhraní k rozšíření WebSphere MQ JMS. Zatímco rozšíření rozhraní JMS produktu WebSphere MQ jsou však specifická pro poskytovatele systému zpráv WebSphere MQ , rozšíření IBM JMS nejsou specifická pro produkt WebSphere MQ a lze ji použít s libovolným poskytovatelem systému zpráv v rámci vrstvené architektury popsané v části “Vrstvená architektura” na stránce 772.

Rozhraní com.ibm.msg.client.wmq.WMQConstants obsahuje definice konstant, které může aplikace použít při nastavení vlastností objektů třídy WebSphere MQ pro objekty JMS pomocí rozšíření IBM JMS. Rozhraní obsahuje konstanty pro poskytovatele systému zpráv WebSphere MQ a konstanty JMS, které jsou nezávislé na libovolném poskytovateli systému zpráv.

Příklady kódu, které následují za předpokladu, že byly spuštěny následující příkazy importu:

```
import com.ibm.msg.client.jms.*;
import com.ibm.msg.client.services.*;
import com.ibm.msg.client.wmq.WMQConstants;
```

Vytvoření továren připojení a míst určení

Než bude aplikace moci vytvářet továrny připojení a cíle pomocí rozšíření IBM JMS, musí nejprve vytvořit objekt továrny JmsFactory. Chcete-li vytvořit objekt továrny JmsFactory, aplikace volá metodu getInstance() třídy továrny JmsFactory, jak je uvedeno v následujícím příkladu:

```
JmsFactoryFactory ff = JmsFactoryFactory.getInstance(JmsConstants.WMQ_PROVIDER);
```

Parametr ve volání metody getInstance() je konstanta, která identifikuje poskytovatele systému zpráv produktu WebSphere MQ jako vybraného poskytovatele systému zpráv. Aplikace pak může pomocí objektu továrny JmsFactoryvytvářet továrny připojení a místa určení.

Chcete-li vytvořit továrnu na připojení, aplikace volá metodu createConnectionFactory () objektu továrny JmsFactory, jak je uvedeno v následujícím příkladu:

```
JmsConnectionFactory factory = ff.createConnectionFactory();
```

Tento příkaz vytvoří objekt továrny JmsConnections výchozími hodnotami pro všechny jeho vlastnosti, což znamená, že se aplikace připojuje k výchozímu správci front v režimu vazeb. Chcete-li, aby se aplikace připojovala v režimu klienta nebo se připojila k jinému správci front, než je výchozí správce front, musí aplikace před vytvořením připojení nastavit příslušné vlastnosti objektu továrny JmsConnection. Další informace o tom, jak to provést, viz “Nastavení vlastností tříd produktu WebSphere MQ pro objekty platformy JMS” na stránce 839.

Třída továrny JmsFactorytaké obsahuje metody pro vytvoření továren na připojení následujících typů:

- JmsQueueConnectionFactory
- JmsTopicConnectionFactory
- Továrna JmsXAConnection
- JmsXAQueueConnectionFactory
- JmsXATopicConnectionFactory

Chcete-li vytvořit objekt fronty, volá aplikace metodu createQueue() objektu továrny JmsFactory, jak je uvedeno v následujícím příkladu:

```
JmsQueue q1 = ff.createQueue("Q1");
```

Tento příkaz vytvoří objekt JmsQueue s výchozími hodnotami pro všechny jeho vlastnosti. Objekt představuje frontu WebSphere MQ s názvem Q1 , která patří do lokálního správce front. Tato fronta může být lokální fronta, alias fronta nebo definice vzdálené fronty.

Metoda createQueue() může také přijmout identifikátor URI (Uniform Resource Identifier) fronty jako parametr. Identifikátor URI fronty je řetězec, který určuje název fronty produktu WebSphere MQ a volitelně název správce front, který je vlastníkem fronty, a jednu či více vlastností objektu JmsQueue . Následující příkaz obsahuje příklad identifikátoru URI fronty:

```
JmsQueue q2 = ff.createQueue("queue://QM2/Q2?persistence=2&priority=5");
```

Objekt JmsQueue vytvořený tímto příkazem reprezentuje frontu produktu WebSphere MQ s názvem Q2 , kterou vlastní správce front QM2, a všechny zprávy odeslané do tohoto místa určení jsou trvalé a mají prioritu 5. Další informace o identifikátorech URI front viz “Uniform resource identifiers (URI)” na stránce 850. Alternativním způsobem nastavení vlastností objektu JmsQueue naleznete v tématu “Nastavení vlastností tříd produktu WebSphere MQ pro objekty platformy JMS” na stránce 839.

Chcete-li vytvořit objekt Topic, může aplikace použít metodu createTopic() objektu továrny JmsFactory, jak je uvedeno v následujícím příkladu:

```
JmsTopic t1 = ff.createTopic("Sport/Football/Results");
```

Tento příkaz vytvoří objekt JmsTopic s výchozími hodnotami pro všechny jeho vlastnosti. Objekt představuje téma s názvem Sport/Football/Results.

Metoda createTopic() může také přijmout URI tématu jako parametr. Identifikátor URI tématu je řetězec, který uvádí název tématu a volitelně jednu nebo více vlastností objektu JmsTopic . Následující příkazy obsahují příklad identifikátoru URI tématu:

```
String s1 = "topic://Sport/Tennis/Results?persistence=1&priority=0";
JmsTopic t2 = ff.createTopic(s1);
```

Objekt JmsTopic vytvořený těmito příkazy představuje téma s názvem Sport/Tennis/Results a všechny zprávy odeslané do tohoto místa určení jsou dočasné a mají prioritu 0. Další informace o identifikátorech URI tématu viz “Uniform resource identifiers (URI)” na stránce 850. Alternativním způsobem nastavení vlastností objektu JmsTopic naleznete v tématu “Nastavení vlastností tříd produktu WebSphere MQ pro objekty platformy JMS” na stránce 839.

Poté, co aplikace vytvořila továrnu připojení nebo místo určení, lze tento objekt použít pouze s vybraným poskytovatelem systému zpráv.

Nastavení vlastností tříd produktu WebSphere MQ pro objekty platformy JMS

Chcete-li nastavit vlastnosti tříd produktu WebSphere MQ pro objekty JMS pomocí rozšíření IBM JMS, aplikace použije metody rozhraní com.ibm.msg.client.JmsPropertyContext .

Pro každý datový typ Java obsahuje kontextové rozhraní JmsPropertymetodu pro nastavení hodnoty vlastnosti s daným datovým typem a metodu pro získání hodnoty vlastnosti s daným datovým typem. Aplikace volá například metodu setIntProperty () k nastavení vlastnosti s celočíselnou hodnotou a volá metodu getIntProperty () k získání vlastnosti s celočíselnou hodnotou.

Instance tříd v balíku com.ibm.mq.jms také dědí metody rozhraní kontextu JmsProperty. Aplikace může proto tyto metody použít k nastavení vlastností objektů MQConnectionFactory, MQQueue a MQTopic.

Když aplikace vytvoří třídy produktu WebSphere MQ pro objekt JMS, jsou všechny vlastnosti s výchozími hodnotami nastaveny automaticky. Když aplikace nastaví vlastnost, nová hodnota nahradí jakoukoli předchozí hodnotu, kterou vlastnost měla. Po nastavení vlastnosti nelze tuto vlastnost odstranit, ale její hodnotu lze změnit.

Pokusí-li se aplikace o nastavení vlastnosti na hodnotu, která není platnou hodnotou vlastnosti, třídy WebSphere MQ pro platformu JMS vyvolá výjimku JMSException. Pokusí-li se aplikace o získání vlastnosti, která nebyla nastavena, je toto chování popsáno ve specifikaci JMS. Třídy WebSphere MQ pro rozhraní JMS vrací výjimku NumberFormatvýjimky pro primitivní datové typy a vrací hodnotu null pro odkazované datové typy.

Kromě předdefinovaných vlastností tříd produktu WebSphere MQ pro objekt JMS může aplikace nastavit vlastní vlastnosti. Tyto vlastnosti definované aplikací jsou ignorovány třídami produktu WebSphere MQ pro platformu JMS.

Další informace o vlastnostech tříd produktu WebSphere MQ pro objekty JMS naleznete v tématu Vlastnosti objektů IBM WebSphere MQ classes for JMS.

Následující kód je příkladem, jak nastavit vlastnosti pomocí rozšíření IBM JMS. Kód nastaví pět vlastností továrny připojení.

```
factory.setIntProperty(WMQConstants.WMQ_CONNECTION_MODE,
                       WMQConstants.WMQ_CM_CLIENT);
factory.setStringProperty(WMQConstants.WMQ_QUEUE_MANAGER, "QM1");
factory.setStringProperty(WMQConstants.WMQ_HOST_NAME, "HOST1");
factory.setIntProperty(WMQConstants.WMQ_PORT, 1415);
factory.setStringProperty(WMQConstants.WMQ_CHANNEL, "QM1.SVR");
factory.setStringProperty(WMQConstants.WMQ_APPLICATIONNAME, "My Application");
```

Výsledkem nastavení těchto vlastností je to, že se aplikace připojuje ke správci front QM1 v režimu klienta pomocí kanálu MQI s názvem QM1.SVR. Správce front je spuštěn v systému s názvem hostitele HOST1a modul listener pro správce front naslouchá na portu číslo 1415. Toto připojení a další připojení správce front přidružená k relacím pod ním mají název aplikace "Moje aplikace" přidružená k těmto relacím.

Poznámka: Správci front spuštěnými na platformách z/OS nepodporují nastavení názvů aplikací, a toto nastavení je proto ignorováno.

Kontextové rozhraní JmsPropertyobsahuje také metodu setObjectProperty (), kterou může aplikace použít k nastavení vlastností. Druhým parametrem metody je objekt, který zapouzdřuje hodnotu vlastnosti. Například následující kód vytvoří celočíselný objekt, který zapouzdří celé číslo 1415, a pak vyvolá funkci setObjectProperty () k nastavení vlastnosti PORT továrny na připojení na hodnotu 1415:

```
Integer port = new Integer(1415);
factory.setObjectProperty(WMQConstants.WMQ_PORT, port);
```

Tento kód je proto ekvivalentem následujícího příkazu:

```
factory.setIntProperty(WMQConstants.WMQ_PORT, 1415);
```

Obráceně metoda getObjectProperty () naopak vrací objekt, který zapouzdřuje hodnotu vlastnosti.

Implicitní převod hodnoty vlastnosti z jednoho datového typu na jiný.

Když aplikace používá metodu JmsPropertykontextu k nastavení nebo získání vlastnosti tříd WebSphere MQ pro objekt JMS, lze hodnotu této vlastnosti implicitně převést z jednoho datového typu na jiný.

Následující příkaz například nastavuje vlastnost PRIORITY objektu JmsQueue q1:

```
q1.setStringProperty(WMQConstants.WMQ_PRIORITY, "5");
```

Vlastnost PRIORITY má celočíselnou hodnotu a proto volání funkce setString() implicitně převede řetězec "5" (zdrojová hodnota) na celočíselnou hodnotu 5 (cílovou hodnotu), která se pak stane hodnotou vlastnosti PRIORITY.

Naopak, následující příkaz získá vlastnost PRIORITY objektu JmsQueue q1:

```
String s1 = q1.getStringProperty(WMQConstants.WMQ_PRIORITY);
```

Celočíselná hodnota 5 (zdrojová hodnota), která je hodnotou vlastnosti PRIORITY, je implicitně převedena na řetězec "5" (cílová hodnota) voláním funkce getStringProperty ().

Převody podporované třídami WebSphere MQ pro JMS jsou zobrazeny v Tabulka 122 na stránce 840.

Tabulka 122. Podporované konverze z jednoho datového typu do jiného

Zdrojový datový typ	**Podporované cílové datové typy**
typ boolean	Řetězec
bajt	int, long, short, String
ZNAK	Řetězec
dvojitý	Řetězec
float	dvojitý, String
celé číslo	dlouhý, řetězec
long	Řetězec
short	int, long, String
Řetězec	logická hodnota, byte, double, float, int, long, short

Obecná pravidla týkající se podporovaných převodů jsou následující:

- Číselné hodnoty lze převádět z jednoho datového typu na jiný za předpokladu, že během převodu nebudou ztracena žádná data. Např. hodnota s datovým typem `int` může být převedena na hodnotu s datovým typem `long`, ale nemůže být převedena na hodnotu s datovým typem `short`.
- Hodnota libovolného datového typu může být převedena na řetězec.

- Řetězec může být převeden na hodnotu libovolného jiného datového typu (kromě `char`), za předpokladu, že řetězec je ve správném formátu pro převod. Pokusí-li se aplikace o převod řetězce, který není ve správném formátu, třídy WebSphere MQ for JMS vrátí výjimku NumberFormatException.
- Pokud se aplikace pokusí o převod, který není podporován, třídy WebSphere MQ pro rozhraní JMS zobrazí výjimku MessageFormatException.

Specifická pravidla pro převod hodnoty z jednoho datového typu do jiného jsou následující:

- Při převodu logické hodnoty na řetězec je hodnota `true` převedena na řetězec "true"a hodnota `false` se převede na řetězec "false".
- Při převodu řetězce na logickou hodnotu je řetězec "true" (bez rozlišování velkých a malých písmen) převeden na `true`a řetězec "false" (bez rozlišování velkých a malých písmen) se převede na `false`. Jakýkoli jiný řetězec se převede na `false`.
- Při převodu řetězce na hodnotu s datovým typem `byte`, `int`, `long`nebo `short`musí řetězec obsahovat následující formát:

  [*mezery*] [*znak*]*číslice*

  Význam komponent řetězce je následující:

  mezery
  Volitelné úvodní prázdné znaky.

  SIGN
  Volitelné znaménko plus (+) nebo znaménko minus (-).

  číslice
  Souvislá posloupnost číslic (0-9). Musí být přítomna alespoň jedna číslice.

  Po posloupnosti číslic může řetězec obsahovat jiné znaky, které nejsou číslice, ale konverze se zastaví, jakmile se dosáhne prvního z těchto znaků. Předpokládá se, že řetězec představuje desítkové celé číslo.

  Pokud řetězec není ve správném formátu, třídy WebSphere MQ pro rozhraní JMS zobrazí výjimku NumberFormatException.
- Při převodu řetězce na hodnotu s datovým typem `double` nebo `float`musí řetězec obsahovat následující formát:

  [*mezery*] [*znak*]*číslice*[*ne_znak*[*_znak*]*_číslice*]

  Význam komponent řetězce je následující:

  mezery
  Volitelné úvodní prázdné znaky.

  SIGN
  Volitelné znaménko plus (+) nebo znaménko minus (-).

  číslice
  Souvislá posloupnost číslic (0-9). Musí být přítomna alespoň jedna číslice.

  _char
  Znak exponent, který je buď *E* , nebo *e*.

  _sign
  Volitelný znak plus (+) nebo minus (-) pro exponent.

  _číslice
  Souvislá posloupnost číslic (0-9) pro exponent. Pokud řetězec obsahuje znak exponent, musí být přítomna alespoň jedna číslice.

  Po posloupnosti číslic nebo volitelných znaků představujících exponent může řetězec obsahovat jiné znaky, které nejsou číslice, ale konverze se zastaví, jakmile se dosáhne prvního z těchto znaků. Předpokládá se, že řetězec představuje desetinné číslo s plovoucí řádovou čárkou s exponentem, který je mocninou 10.

  Pokud řetězec není ve správném formátu, třídy WebSphere MQ pro rozhraní JMS zobrazí výjimku NumberFormatException.

- Při převodu číselné hodnoty (včetně hodnoty s datovým typem byte) na řetězec se hodnota převede na řetězcovou reprezentaci hodnoty jako dekadické číslo, nikoli řetězec obsahující znak ASCII pro danou hodnotu. Například celé číslo 65 se převede na řetězec "65", nikoli na řetězec "A".

Nastavení více než jedné vlastnosti v jednom volání

Rozhraní kontextu JmsPropertytaké obsahuje metodu setBatchProperties (), kterou může aplikace použít k nastavení více než jedné vlastnosti v rámci jednoho volání. Parametr metody je objekt mapy, který zapouzdřuje sadu dvojic název-hodnota-hodnota.

Následující kód například používá metodu Properties () setBatchk nastavení stejných pěti vlastností továrny připojení, jak ukazuje “Nastavení vlastností tříd produktu WebSphere MQ pro objekty platformy JMS” na stránce 839. Kód vytvoří instanci třídy HashMap , která implementuje rozhraní Map.

```
HashMap batchProperties = new HashMap();
batchProperties.put(WMQConstants.WMQ_CONNECTION_MODE,
                    new Integer(WMQConstants.WMQ_CM_CLIENT));
batchProperties.put(WMQConstants.WMQ_QUEUE_MANAGER, "QM1");
batchProperties.put(WMQConstants.WMQ_WMQ_HOST_NAME, "HOST1");
batchProperties.put(WMQConstants.WMQ_PORT, "1414");
batchProperties.put(WMQConstants.WMQ_CHANNEL, "QM1.SVR");
factory.setBatchProperties(batchProperties);
```

Všimněte si, že druhý parametr metody Map.put() musí být objekt. Proto musí být hodnota vlastnosti s primitivním datovým typem uzavřena v rámci objektu nebo znázorněna řetězcem, jak je uvedeno v příkladu.

Metoda setBatchProperties () ověřuje každou vlastnost. Pokud metoda Properties () setBatchnemůže nastavit vlastnost, protože její hodnota například není platná, není nastavena žádná z uvedených vlastností.

Názvy a hodnoty vlastností

Pokud aplikace používá metody rozhraní kontextu JmsPropertyk nastavení a získání vlastností tříd WebSphere MQ pro objekty platformy JMS, může aplikace určit názvy a hodnoty vlastností kterýmikoli z následujících způsobů. Každý z doprovodných příkladů uvádí, jak nastavit vlastnost PRIORITY objektu JmsQueue objektu q1 tak, aby zpráva odeslaná do fronty měla prioritu určenou ve volání metody send ().

Pomocí názvů vlastností a hodnot, které jsou definovány jako konstanty v rozhraní com.ibm.msg.client.wmq.WMQConstants

Následující příkaz je příkladem toho, jak lze určit názvy a hodnoty vlastností tímto způsobem:

```
q1.setIntProperty(WMQConstants.WMQ_PRIORITY, WMQConstants.WMQ_PRI_APP);
```

Použití názvů a hodnot vlastností, které lze použít ve frontě a v uniformných identifikátorech prostředku (URI)

Následující příkaz je příkladem toho, jak lze určit názvy a hodnoty vlastností tímto způsobem:

```
q1.setIntProperty("priority", -2);
```

Do tohoto způsobu lze zadat pouze názvy a hodnoty vlastností cílů.

Použití názvů a hodnot vlastností, které jsou rozpoznány nástrojem pro administraci produktu WebSphere MQ JMS

Následující příkaz je příkladem toho, jak lze určit názvy a hodnoty vlastností tímto způsobem:

```
q1.setStringProperty("PRIORITY", "APP");
```

Zkrácený tvar názvu vlastnosti je také přijatelný, jak je uvedeno v následujícím příkazu:

```
q1.setStringProperty("PRI", "APP");
```

Když aplikace získá vlastnost, vrácená hodnota závisí na způsobu, jakým aplikace uvádí název vlastnosti. Například, pokud aplikace uvádí konstantu WMQConstants.WMQ_PRIORITY jako název vlastnosti, vrácená hodnota je celé číslo -2:

```
int n1 = getIntProperty(WMQConstants.WMQ_PRIORITY);
```

Stejná hodnota je vrácena, pokud aplikace uvádí řetězec "priority" jako název vlastnosti:

```
int n2 = getIntProperty("priority");
```

Pokud však aplikace specifikuje řetězec "PRIORITY" nebo "PRI" jako název vlastnosti, vrácená hodnota je řetězec "APP":

```
String s1 = getStringProperty("PRI");
```

Interně třídy produktu WebSphere MQ pro systém JMS ukládají názvy a hodnoty vlastností jako literálové hodnoty definované v rozhraní com.ibm.msg.client.wmq.WMQConstants . Jedná se o definovaný kanonický formát pro názvy vlastností a hodnoty. Obecným pravidlem je, že pokud aplikace nastavuje vlastnosti pomocí jednoho z dalších dvou způsobů zadání názvů a hodnot vlastností, třídy WebSphere MQ pro JMS musí převádět názvy a hodnoty z uvedeného vstupního formátu do kanonického formátu. Podobně platí, že pokud aplikace získá vlastnosti pomocí jednoho z dalších dvou způsobů zadání názvů a hodnot vlastností, třídy WebSphere MQ pro platformu JMS musí převést názvy z určeného vstupního formátu do kanonického formátu a převádět hodnoty z kanonického formátu do požadovaného výstupního formátu. Provedení těchto převodů může mít vliv na výkon.

Názvy vlastností a hodnoty vrácené výjimkami, v souborech trasování nebo v třídách WebSphere MQ pro protokol JMS jsou vždy v kanonickém formátu.

Použití rozhraní Map

Rozhraní kontextu JmsPropertyrozšiřuje rozhraní java.util.Map . Aplikace proto může používat metody rozhraní Map k přístupu k vlastnostem objektů WebSphere MQ pro objekt JMS.

Následující kód například vytiskne názvy a hodnoty všech vlastností továrny připojení. Kód používá pouze metody mapových rozhraní k získání názvů a hodnot vlastností.

```
// Get the names of all the properties
Set propNames = factory.keySet();

// Loop round all the property names and get the property values
Iterator iterator = propNames.iterator();
while (iterator.hasNext()){
    String pName = (String)iterator.next();
    System.out.println(pName+"="+factory.get(pName));
}
```

Použití metod Map interface neobejdete žádné ověření platnosti vlastností ani konverze.

Použití rozšíření WebSphere MQ JMS

Třídy WebSphere MQ pro službu JMS obsahují sadu rozšíření rozhraní JMS API s názvem WebSphere MQ JMS. Aplikace může tato rozšíření použít k dynamickému vytváření továren připojení a cílů v běhovém prostředí a k nastavení vlastností továren připojení a cílů.

Třídy WebSphere MQ pro službu JMS obsahují sadu tříd v balících com.ibm.jms a com.ibm.mq.jms. Tyto třídy implementují rozhraní JMS a obsahují rozšíření produktu WebSphere MQ JMS. Příklady kódu, které následují za předpokladu, že tyto balíky byly importovány následujícími příkazy:

```
import com.ibm.jms.*;
import com.ibm.mq.jms.*;
```

Aplikace může používat rozšíření produktu WebSphere MQ JMS k provádění následujících funkcí:

- Dynamicky vytvářet továrny připojení a místa určení za běhu namísto jejich načtení jako spravované objekty z oboru názvů rozhraní JNDI (Java Naming and Directory Interface)

- Nastavit vlastnosti továren připojení a cílů

Vytváření továren na připojení

Chcete-li vytvořit továrnu na připojení, může aplikace použít konstruktor MQConnectionFactory , jak ukazuje následující příklad:

```
MQConnectionFactory factory = new MQConnectionFactory();
```

Tento příkaz vytvoří objekt MQConnectionFactory s výchozími hodnotami pro všechny jeho vlastnosti, což znamená, že se aplikace připojí k výchozímu správci front v režimu vazeb. Chcete-li, aby se aplikace připojovala v režimu klienta nebo se připojila k jinému správci front, než je výchozí správce front, musí aplikace před vytvořením připojení nastavit odpovídající vlastnosti objektu MQConnectionFactory . Další informace o tom, jak to provést, viz “Nastavení vlastností továren připojení” na stránce 844.

Aplikace může vytvořit továrny připojení následujících typů podobným způsobem:

- Továrna MQQueueConnection
- Továrna MQTopicConnection
- MQXAConnectionFactory
- Továrna MQXAQueueConnection
- Továrna MQXATopicConnection

Nastavení vlastností továren připojení

Aplikace může nastavit vlastnosti továrny připojení voláním příslušných metod továrny připojení. Továrnu připojení může buď být spravovaný objekt, nebo objekt vytvořený dynamicky za běhu.

Prohlédněte si následující kód, například:

```
MQConnectionFactory factory = new MQConnectionFactory();
.
factory.setTransportType(WMQConstants.WMQ_CM_CLIENT);
factory.setQueueManager("QM1");
factory.setHostName("HOST1");
factory.setPort(1415);
factory.setChannel("QM1.SVR");
```

Tento kód vytvoří objekt MQConnectionFactory a poté nastaví pět vlastností objektu. Výsledkem nastavení těchto vlastností je to, že se aplikace připojuje ke správci front QM1 v režimu klienta pomocí kanálu MQI s názvem QM1.SVR. Správce front je spuštěn v systému s názvem hostitele HOST1a modul listener pro správce front naslouchá na portu číslo 1415.

V případě připojení v reálném čase ke zprostředkovateli může aplikace použít následující kód:

```
MQConnectionFactory factory = new MQConnectionFactory();
.
factory.setTransportType(WMQConstants.WMQ_CM_DIRECT);
factory.setHostName("HOST2");
factory.setPort(1507);
```

Tento kód předpokládá, že zprostředkovatel běží na systému s názvem hostitele HOST2 a naslouchá na portu číslo 1507.

Aplikace, která používá v reálném čase připojení k zprostředkovateli, může používat pouze styl publikování/odběru zpráv. Nemůže používat styl systému zpráv typu point-to-point.

Platné jsou pouze určité kombinace vlastností továrny připojení. Informace o tom, které kombinace jsou platné, najdete v tématu Závislosti mezi vlastnostmi tříd produktu WebSphere MQ pro objekty platformy JMS .

Další informace o vlastnostech továrny připojení a metodách používaných k nastavení jejích vlastností naleznete v tématu Vlastnosti objektů IBM WebSphere MQ classes for JMS.

Vytváření cílů

Chcete-li vytvořit objekt fronty, může aplikace použít konstruktor MQQueue, jak je uvedeno v následujícím příkladu:

```
MQQueue q1 = new MQQueue("Q1");
```

Tento příkaz vytvoří objekt MQQueue s výchozími hodnotami pro všechny jeho vlastnosti. Objekt představuje frontu WebSphere MQ s názvem Q1 , která patří do lokálního správce front. Tato fronta může být lokální fronta, alias fronta nebo definice vzdálené fronty.

Alternativní formulář konstruktoru MQQueue má dva parametry, jak je zobrazeno v následujícím příkladu:

```
MQQueue q2 = new MQQueue("QM2", "Q2");
```

Objekt MQQueue vytvořený tímto příkazem reprezentuje frontu produktu WebSphere MQ s názvem Q2 , kterou vlastní správce front QM2. Identifikovaným správcem front může být lokální správce front nebo vzdálený správce front. Pokud se jedná o vzdáleného správce front, musí být produkt WebSphere MQ nakonfigurován tak, aby v případě, že aplikace odešle zprávu do tohoto místa určení, může produkt Websphere MQ směrovat zprávu z lokálního správce front do vzdáleného správce front.

Konstruktor MQQueue může také přijmout identifikátor URI (Uniform Resource Identifier) fronty jako jeden parametr. Identifikátor URI fronty je řetězec, který určuje název fronty produktu WebSphere MQ a volitelně název správce front, který je vlastníkem fronty, a jednu či více vlastností objektu MQQueue. Následující příkaz obsahuje příklad identifikátoru URI fronty:

```
MQQueue q3 = new MQQueue("queue://QM3/Q3?persistence=2&priority=5");
```

Objekt MQQueue vytvořený tímto příkazem reprezentuje frontu WebSphere 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 front viz “Uniform resource identifiers (URI)” na stránce 850. Alternativním způsobem nastavení vlastností objektu MQQueue naleznete v tématu “Nastavení vlastností cílů” na stránce 845.

Chcete-li vytvořit objekt Topic, může aplikace použít konstruktor MQTopic, jak je uvedeno v následujícím příkladu:

```
MQTopic t1 = new MQTopic("Sport/Football/Results");
```

Tento příkaz vytvoří objekt MQTopic s výchozími hodnotami pro všechny jeho vlastnosti. Objekt představuje téma s názvem Sport/Football/Results.

Konstruktor MQTopic může také přijmout identifikátor URI tématu jako parametr. Identifikátor URI tématu je řetězec, který určuje název tématu a volitelně také jednu nebo více vlastností objektu MQTopic. Následující příkaz obsahuje příklad identifikátoru URI tématu:

```
MQTopic t2 = new MQTopic("topic://Sport/Tennis/Results?persistence=1&priority=0");
```

Objekt MQTopic vytvořený tímto příkazem představuje téma s názvem Sport/Tennis/Results a všechny zprávy odeslané do tohoto místa určení jsou dočasné a mají prioritu 0. Další informace o identifikátorech URI tématu viz “Uniform resource identifiers (URI)” na stránce 850. Alternativním způsobem nastavení vlastností objektu MQTopic naleznete v tématu “Nastavení vlastností cílů” na stránce 845.

Nastavení vlastností cílů

Aplikace může nastavit vlastnosti cíle voláním odpovídajících metod cíle. Místo určení může být buď administrovaný objekt, nebo objekt vytvořený dynamicky za běhu.

Prohlédněte si následující kód, například:

```
MQQueue q1 = new MQQueue("Q1");
.
q1.setPersistence(WMQConstants.WMQ_PER_PER);
q1.setPriority(5);
```

Tento kód vytvoří objekt MQQueue a poté nastaví dvě vlastnosti objektu. Výsledkem nastavení těchto vlastností je to, že všechny zprávy odeslané do místa určení jsou trvalé a mají prioritu 5.

Aplikace může nastavit vlastnosti objektu MQTopic podobným způsobem, jak je uvedeno v následujícím příkladu:

```
MQTopic t1 = new MQTopic("Sport/Football/Results");
.
t1.setPersistence(WMQConstants.WMQ_PER_NON);
t1.setPriority(0);
```

Tento kód vytvoří objekt MQTopic a poté nastaví dvě vlastnosti objektu. Výsledkem nastavení těchto vlastností je, že všechny zprávy odeslané do místa určení jsou netrvalé a mají prioritu 0.

Další informace o vlastnostech cíle a metodách použitých k nastavení jejích vlastností naleznete v tématu Vlastnosti objektů IBM WebSphere MQ classes for JMS.

Sestavení připojení v aplikaci JMS

Chcete-li vytvořit připojení, použije aplikace JMS objekt ConnectionFactory k vytvoření objektu připojení a pak spustí připojení.

Chcete-li vytvořit objekt připojení, aplikace použije metodu createConnection() objektu ConnectionFactory , jak je uvedeno v následujícím příkladu:

```
ConnectionFactory factory;
Connection connection;
.
.
.
connection = factory.createConnection();
```

Při vytvoření připojení JMS vytvoří IBM WebSphere MQ classes for JMS popisovač připojení (Hconn) a zahájí konverzaci se správcem front.

Rozhraní továrny QueueConnectiona rozhraní továrny TopicConnectioneach dědí metodu createConnection() z rozhraní ConnectionFactory . Proto můžete použít metodu createConnection() k vytvoření objektu specifického pro doménu, jak je zobrazeno v následujícím příkladu:

```
QueueConnectionFactory qcf;
Connection connection;
.
.
.
connection = qcf.createConnection();
```

Tento fragment kódu vytvoří objekt QueueConnection . Aplikace může nyní na tomto objektu provést nezávislou operaci na doméně, nebo operaci, která je použitelná pouze pro doménu typu point-to-point. Pokud se však aplikace pokusí provést operaci, která je použitelná pouze pro doménu publikování/odběru, vyvolá se výjimka výjimky IllegalStates touto zprávou:

```
JMSMQ1112: Operation for a domain specific object was not valid.
           Operation createProducer() is not valid for type com.ibm.mq.jms.MQTopic
```

Důvodem je to, že připojení bylo vytvořeno z továrny připojení specifické pro doménu.

Poznámka: Všimněte si, že ID procesu aplikace se používá jako výchozí identita uživatele, která má být předána správci front. Je-li aplikace spuštěna v režimu transportu klienta, musí toto ID procesu existovat spolu s příslušnými oprávněními na serveru. Chcete-li použít jinou identitu, použijte metodu createConnection(jméno uživatele, heslo).

Specifikace JMS uvádí, že připojení je vytvořeno ve stavu `stopped` . Do zahájení připojení nemůže spotřebitel zpráv, který je přidružen k tomuto připojení, přijímat žádné zprávy. Chcete-li spustit připojení, aplikace použije metodu start () objektu připojení, jak je zobrazeno v následujícím příkladu:

```
connection.start();
```

Vytvoření relace v aplikaci JMS

K vytvoření relace používá aplikace platformy JMS metodu createSession() objektu připojení.

Metoda createSession() má dva parametry:

1. Parametr, který určuje, zda relace obsahuje transakci, nebo nikoli.
2. Parametr, který uvádí režim potvrzení pro relaci

Například následující kód vytvoří relaci, která se neobchoduje a má režim potvrzení AUTO_ACKNOWLEDGE:

```
Session session;
.
boolean transacted = false;
session = connection.createSession(transacted, Session.AUTO_ACKNOWLEDGE);
```

Při vytvoření relace JMS vytvoří produkt IBM WebSphere MQ classes for JMS popisovač připojení (Hconn) a zahájí konverzaci se správcem front.

Objekt relace a objekt MessageProducer nebo MessageConsumer z něj vytvořené nemohou být souběžně používány různými podprocesy aplikace s podporou podprocesů. Nejjednodušším způsobem, jak zajistit, aby tyto objekty nebyly použity souběžně, je vytvoření samostatného objektu relace pro každý podproces.

Provedené relace v aplikacích JMS

Aplikace JMS mohou spouštět lokální transakce při prvním vytvoření relace transakce. Aplikace může potvrdit nebo odvolat transakci.

Aplikace platformy JMS mohou spouštět lokální transakce. Lokální transakce je transakce, která zahrnuje změny pouze na prostředky správce front, ke kterému je aplikace připojena. Chcete-li spustit lokální transakce, musí aplikace nejprve vytvořit relaci transakce voláním metody createSession() objektu připojení, která určuje jako parametr, že relace je zpracovávána. Následně jsou všechny zprávy odeslané a přijaté v rámci relace seskupeny do posloupnosti transakcí. Transakce je ukončena, když aplikace potvrdí nebo odvolá zprávy, které odeslala a obdržela od začátku transakce.

Pro potvrzení transakce volá aplikace metodu commit () objektu Session. Když je transakce potvrzena, všechny zprávy odeslané v rámci transakce se stanou dostupnými pro doručení do jiných aplikací a všechny zprávy přijaté v rámci transakce jsou potvrzeny, aby se server systému zpráv nepokoušel o jejich doručení do aplikace znovu. V doméně typu point-to-point server zpráv také odebírá přijaté zprávy z jejich front.

Chcete-li odvolat transakci, volá aplikace metodu rollback () objektu Session. Když je transakce odvolána, všechny zprávy odeslané v rámci transakce jsou vyřazeny serverem systému zpráv a všechny zprávy přijaté v rámci transakce budou znovu k dispozici pro doručení. V dvoubodové doméně se zprávy, které byly přijaty, vrátí zpět do front a znovu se stanou viditelnými pro jiné aplikace.

Nová transakce se spustí automaticky, když aplikace vytvoří transakci s transakcemi nebo zavolá metodu commit () nebo rollback (). Proto má relace, která obsahuje transakci, vždy aktivní transakci.

Když aplikace zavře relaci transakce, dojde k implicitnímu odvolání transakce. Když aplikace uzavře připojení, dojde k implicitnímu odvolání transakce pro všechny relace, které se v relaci nachází.

Pokud aplikace skončí bez zavření připojení, dojde také k implicitnímu odvolání transakce pro všechny relace obsažené v transakci, které se týká.

Transakce je zcela obsažena v rámci relace transakce. Transakce nemůže přesahovat relace. To znamená, že aplikace nemůže odesílat a přijímat zprávy ve dvou nebo více transakčních relacích, a poté potvrdit nebo odvolat všechny tyto akce jako jedinou transakci.

Režimy potvrzení relací JMS

Každá relace, která nemá transakci, má režim potvrzení, který určuje, jak jsou přijímány zprávy přijaté aplikací. K dispozici jsou tři režimy potvrzení a volba režimu potvrzení má vliv na návrh aplikace.

Pokud relace není součástí transakce, je způsob, jakým zprávy přijaté aplikací potvrdí, je určen režimem potvrzení relace. Tři režimy potvrzení jsou popsány v následujících odstavcích:

AUTOMATICKÉ_POTVRZENÍ

Relace automaticky potvrdí každou zprávu přijatou aplikací.

Pokud jsou zprávy doručovány synchronně do aplikace, relace potvrdí příjem zprávy pokaždé, když je úspěšně dokončeno přijetí volání. Pokud jsou zprávy doručovány asynchronně, relace potvrdí přijetí zprávy pokaždé, když se úspěšně dokončí volání metody onMessage() modulu listener pro zprávy.

Pokud aplikace obdrží zprávu úspěšně, ale selhání zabrání provedení potvrzení, bude zpráva znovu k dispozici pro doručení. Aplikace musí proto být schopna zpracovat znovu dodanou zprávu.

PUP_OK_ACKNOWLEDGE

Relace bere na vědomí zprávy přijaté aplikací v době, kdy je vybrána.

Použití tohoto režimu potvrzení snižuje množství práce, které musí relace dělat, ale selhání zabraňující potvrzení zprávy může vést k tomu, že bude znovu k dispozici více než jedna zpráva k doručení. Aplikace musí být proto schopna zpracovávat zprávy, které jsou znovu doručeny.

Omezení: V režimech AUTO_ACKNOWLEDGE a DUPS_OK_ACKNOWLEDGE nepodporuje rozhraní JMS aplikaci zahazování neošetřené výjimky v modulu listener pro zprávy. To znamená, že zprávy jsou vždy potvrzeny, když se modul listener pro zprávy vrátí, bez ohledu na to, zda byl úspěšně zpracován (za předpokladu, že došlo k neúspěchům selhání) a nezabráníte tomu, aby aplikace pokračovala). Pokud vyžadujete lepší kontrolu potvrzení zprávy, použijte režimy CLIENT_ACKNOWLEDGE nebo transakci, které poskytují aplikaci plnou kontrolu nad funkcemi potvrzení.

CLIENT_ACKNOWLEDGE

Aplikace potvrdí zprávy, které obdrží, voláním metody Acknowledge třídy Message.

Aplikace může potvrdit příjem každé zprávy jednotlivě, nebo může obdržet dávku zpráv a zavolat metodu Potvrdit pouze pro poslední zprávu, kterou obdrží. Když se metoda Acknowledge nazývá všechny zprávy přijaté od poslední doby, kdy byla metoda volána, jsou potvrzeny.

Ve spojení s jakýmkoli z těchto režimů potvrzení může aplikace zastavit a znovu spustit doručování zpráv v relaci voláním metody Recover třídy Session. Přijaté zprávy, ale dříve nepotvrzené, jsou znovu doručeny. Je však možné, že nebudou doručeny ve stejné posloupnosti, v jaké byly dříve doručeny. Do té doby mohla dorazila zpráva s vyšší prioritou a některé původní zprávy mohly vypršet. V doméně dvoubodového spojení mohly být některé z původních zpráv spotřebovány jinou aplikací.

Aplikace může určit, zda je zpráva znovu doručena, tím, že zkontrolujete obsah pole záhlaví JMSRedelivered zprávy. Aplikace to provede voláním metody getJMSRedelivered() třídy zpráv.

Vytvoření cílů v aplikaci JMS

Místo načítání cílů jako spravovaných objektů z oboru názvů rozhraní JNDI (Java Naming and Directory Interface) může aplikace platformy JMS použít relaci k dynamickému vytváření cílů za běhu. Aplikace může použít identifikátor URI (Uniform Resource Identifier) k identifikaci fronty nebo tématu produktu WebSphere MQ a volitelně k určení jedné či více vlastností objektu Fronta nebo Téma.

Použití relace k vytvoření objektů fronty

Chcete-li vytvořit objekt fronty, může aplikace použít metodu createQueue() objektu relace, jak je uvedeno v následujícím příkladu:

```
Session session;
.
Queue q1 = session.createQueue("Q1");
```

Tento kód vytvoří objekt fronty s výchozími hodnotami pro všechny jeho vlastnosti. Objekt představuje frontu WebSphere MQ s názvem Q1 , která patří do lokálního správce front. Tato fronta může být lokální fronta, alias fronta nebo definice vzdálené fronty.

Metoda createQueue() také přijímá identifikátor URI fronty jako parametr. Identifikátor URI fronty je řetězec, který určuje název fronty produktu WebSphere MQ a volitelně název správce front, který vlastní

frontu, a jednu či více vlastností objektu Fronta. Následující příkaz obsahuje příklad identifikátoru URI fronty:

```
Queue q2 = session.createQueue("queue://QM2/Q2?persistence=2&priority=5");
```

Objekt fronty vytvořený tímto příkazem reprezentuje frontu WebSphere MQ nazvanou Q2 , kterou vlastní správce front s názvem QM2, a všechny zprávy odeslané do tohoto místa určení jsou trvalé a mají prioritu 5. Identifikovaným správcem front může být lokální správce front nebo vzdálený správce front. Pokud se jedná o vzdáleného správce front, musí být produkt WebSphere MQ nakonfigurován tak, aby v případě, že aplikace odešle zprávu do tohoto místa určení, produkt Websphere MQ může směrovat zprávu z lokálního správce front do správce front QM2. Další informace o identifikátorech URI viz “Uniform resource identifiers (URI)” na stránce 850.

Všimněte si, že parametr na metodě createQueue() obsahuje informace specifické pro poskytovatele. Proto při použití metody createQueue() pro vytvoření objektu Queue místo načtení objektu Queue jako spravovaného objektu z oboru názvů rozhraní JNDI může být vaše aplikace méně přenosná.

Aplikace může vytvořit objekt TemporaryQueue pomocí metody createTemporaryQueue () objektu Session, jak je uvedeno v následujícím příkladu:

```
TemporaryQueue q3 = session.createTemporaryQueue();
```

Přestože se relace používá k vytvoření dočasné fronty, rozsah dočasné fronty je připojení, které bylo použito k vytvoření relace. Kterákoli z relací připojení může vytvořit správce zpráv a spotřebitele zpráv pro dočasnou frontu. Dočasná fronta zůstane až do konce připojení nebo aplikace explicitně odstraní dočasnou frontu pomocí metody .delete () TemporaryQueue.delete (), podle toho, co nastane dříve.

Když aplikace vytvoří dočasnou frontu, třídy WebSphere MQ pro JMS vytvoří dynamickou frontu ve správci front, ke kterému je aplikace připojena. Vlastnost TEMPMODEL továrny připojení uvádí název modelové fronty, která se používá k vytvoření dynamické fronty, a vlastnost TEMPQPREFIX továrny připojení uvádí předponu, která se používá k vytvoření názvu dynamické fronty.

Použití relace k vytvoření objektů tématu

Chcete-li vytvořit objekt Topic, může aplikace použít metodu createTopic() objektu relace, jak je uvedeno v následujícím příkladu:

```
Session session;
.
Topic t1 = session.createTopic("Sport/Football/Results");
```

Tento kód vytvoří objekt tématu s výchozími hodnotami pro všechny jeho vlastnosti. Objekt představuje téma s názvem Sport/Football/Results.

Metoda createTopic() také přijímá identifikátor URI tématu jako parametr. Identifikátor URI tématu je řetězec, který určuje název tématu a volitelně také jednu nebo více vlastností objektu tématu. Následující kód obsahuje příklad identifikátoru URI tématu:

```
String uri = "topic://Sport/Tennis/Results?persistence=1&priority=0";
Topic t2 = session.createTopic(uri);
```

Objekt Topic vytvořený tímto kódem představuje téma s názvem Sport/Tennis/Results a všechny zprávy odeslané do tohoto místa určení jsou dočasné a mají prioritu 0. Další informace o identifikátorech URI tématu viz “Uniform resource identifiers (URI)” na stránce 850.

Všimněte si, že parametr na metodě createTopic() obsahuje informace specifické pro poskytovatele. Proto při použití metody createTopic() k vytvoření objektu Topic místo načtení objektu tématu jako spravovaného objektu z oboru názvů JNDI může být vaše aplikace méně přenosná.

Aplikace může vytvořit objekt TemporaryTopic pomocí metody Topic () objektu relace createTemporary, jak je uvedeno v následujícím příkladu:

```
TemporaryTopic t3 = session.createTemporaryTopic();
```

Ačkoli se relace používá k vytvoření dočasného tématu, rozsah dočasného tématu je připojení, které bylo použito k vytvoření relace. Kterákoli z relací připojení může vytvořit správce zpráv a spotřebitele zpráv pro dočasné téma. Dočasné téma zůstane až do konce připojení nebo aplikace explicitně odstraní dočasné téma pomocí metody .delete () TemporaryTopic.delete (), podle toho, co nastane dříve.

Když aplikace vytvoří dočasné téma, třídy WebSphere MQ pro JMS vytvoří téma s názvem, který začíná se znaky TEMP/*tempTopicPředpona*, kde *tempTopicPředpona* je hodnota vlastnosti TEMPTOPICPREFIX továrny připojení.

Uniform resource identifiers (URI)

Identifikátor URI fronty je řetězec, který určuje název fronty produktu WebSphere MQ a volitelně název správce front, který je vlastníkem fronty, a jednu či více vlastností objektu Queue vytvořeného aplikací. Identifikátor URI tématu je řetězec, který určuje název tématu a volitelně také jednu nebo více vlastností objektu tématu vytvořeného aplikací.

Identifikátor URI fronty má následující formát:

```
queue://[qMgrName]/qName[?propertyName1=propertyValue1
                         &propertyName2=propertyValue2
                         &...]
```

Identifikátor URI tématu má následující formát:

```
topic://topicName[?propertyName1=propertyValue1
                  &propertyName2=propertyValue2
                  &...]
```

Proměnné v těchto formátech mají následující význam:

QmgrName
Název správce front, který vlastní frontu určenou identifikátorem URI.

Správce front může být lokálním správcem front nebo vzdáleným správcem front. Pokud se jedná o vzdáleného správce front, musí být produkt WebSphere MQ nakonfigurován tak, aby při odesílání zprávy do fronty produkt Websphere MQ mohl odeslat zprávu z lokálního správce front do vzdáleného správce front.

Není-li zadán žádný název, předpokládá se lokální správce front.

qName
Název fronty produktu WebSphere MQ .

Fronta může být lokální fronta, alias fronta nebo definice vzdálené fronty.

Pravidla pro vytváření názvů front naleznete v tématu Pravidla pojmenování objektů produktu IBM WebSphere MQ .

topicName
Název tématu.

Pravidla pro vytváření názvů témat najdete v tématu Pravidla pojmenování objektů produktu IBM WebSphere MQ. Vyvarujte se použití zástupných znaků +, #, *, a? v názvech témat. Názvy témat obsahující tyto znaky mohou způsobit neočekávané výsledky při jejich přihlášení k odběru. Viz Použití řetězců témat.

***propertyName1*, *propertyName2*, ...**
Názvy vlastností objektu Queue nebo Topic vytvořeného aplikací. Produkt Tabulka 123 na stránce 851 vypíše platné názvy vlastností, které lze použít v identifikátoru URI.

Nejsou-li zadány žádné vlastnosti, má objekt Fronta nebo Téma výchozí hodnoty pro všechny své vlastnosti.

***propertyValue1*, *propertyValue2*, ...**
Hodnoty vlastností objektu Queue nebo Topic vytvořeného aplikací. Tabulka 123 na stránce 851 zobrazuje seznam platných hodnot vlastností, které lze použít v identifikátoru URI.

Hranaté závorky ([]) označují volitelnou komponentu a tři tečky (...) znamenají, že seznam dvojic název-hodnota, je-li přítomen, může obsahovat jednu nebo více dvojic název-hodnota.

Tabulka 123 na stránce 851 uvádí seznam platných názvů vlastností a platných hodnot, které lze použít ve frontě a identifikátorech URI témat. Ačkoli administrativní nástroj produktu WebSphere MQ JMS používá symbolické konstanty pro hodnoty vlastností, identifikátory URI nemohou obsahovat symbolické konstanty.

Tabulka 123. Názvy vlastností a platné hodnoty pro použití ve frontě a identifikátorech URI témat

Název vlastnosti	**Popis**	**Platné hodnoty**
CCSID	Způsob, jakým jsou znaková data v těle zprávy reprezentována, když třídy WebSphere MQ pro rozhraní JMS předávají zprávu do cíle	• Jakýkoli identifikátor kódované znakové sady podporovaný produktem WebSphere MQ.
kódování	Jak jsou číselná data v těle zprávy představována, když třídy WebSphere MQ pro rozhraní JMS předávají zprávu do cíle	• Libovolná platná hodnota pro pole *Kódování* v deskriptoru zpráv produktu WebSphere MQ .
Vypršení	Doba platnosti pro zprávy odeslané do místa určení	• -2-Jak je uvedeno ve volání send () nebo, pokud není uveden ve volání send (), výchozí doba pro život producenta zpráv. • 0-Platnost zprávy odeslané do místa určení nikdy nevyprší. • Kladné celé číslo určující dobu života v milisekundách.
výběrové vysílání	Nastavení výběrového vysílání pro téma při použití připojení v reálném čase ke zprostředkovateli	Platné hodnoty jsou uvedeny v následujícím seznamu. Přiřazeno ke každé hodnotě je odpovídající hodnota vlastnosti MULTICAST, jak je použita v nástroji pro administraci produktu WebSphere MQ JMS. Popis vlastnosti MULTICAST a jeho platných hodnot naleznete v tématu Vlastnosti objektů IBM WebSphere MQ classes for JMS. • -1-ASCF • 0 - zakázáno • 3-NOTR • 5-SPOLEHLIVÉ • 7-POVOLENO

Tabulka 123. Názvy vlastností a platné hodnoty pro použití ve frontě a identifikátorech URI témat (pokračování)

Název vlastnosti	Popis	Platné hodnoty
trvání, perzistence	Perzistence zpráv odeslaných do místa určení	• -2-Jak je uvedeno ve volání send () nebo, pokud není uveden ve volání send (), výchozí trvání producenta zpráv. • -1-Jak je uvedeno atributem *DefPersistence* fronty nebo tématu WebSphere MQ . • 1-Netrvalý. • 2-Trvalý. • 3-Ekvivalentní k hodnotě HIGH pro vlastnost PERSISTENCE, jak je použita v nástroji pro administraci produktu WebSphere MQ JMS. Vysvětlení této hodnoty najdete v tématu “Trvalé zprávy JMS” na stránce 874.
priority (priorita)	Priorita zpráv odeslaných do místa určení	• -2-Jak je uvedeno ve volání send () nebo, pokud není uvedeno ve volání send (), výchozí priorita producentu zprávy. • -1-Jak je uvedeno atributem *DefPriority* fronty nebo tématu WebSphere MQ . • Celé číslo v rozsahu 0-9 určující prioritu zpráv odeslaných do místa určení.
targetClient	Určuje, zda zprávy odeslané do místa určení obsahují záhlaví MQRFH2 .	• 0-Zprávy obsahují záhlaví MQRFH2 . • 1-Zprávy neobsahují záhlaví MQRFH2 .

Následující identifikátor URI například identifikuje frontu produktu WebSphere MQ s názvem Q1 , kterou vlastní lokální správce front. Objekt fronty vytvořený s použitím tohoto identifikátoru URI má výchozí hodnoty pro všechny jeho vlastnosti.

```
queue:///Q1
```

Následující identifikátor URI identifikuje frontu produktu WebSphere MQ s názvem Q2 , kterou vlastní správce front s názvem QM2. Všechny zprávy odeslané do tohoto místa určení mají prioritu 6. Zbývající vlastnosti objektu Queue vytvořeného pomocí tohoto identifikátoru URI mají své výchozí hodnoty.

```
queue://QM2/Q2?priority=6
```

Následující identifikátor URI identifikuje téma s názvem Sport/Athletics/Results. Všechny zprávy odeslané do tohoto místa určení jsou netrvalé a mají prioritu 0. Zbývající vlastnosti objektu Topic vytvořeného pomocí tohoto identifikátoru URI mají své výchozí hodnoty.

```
topic://Sport/Athletics/Results?persistence=1&priority=0
```

Odesílání zpráv v aplikaci JMS

Než může aplikace JMS odesílat zprávy do místa určení, musí nejprve vytvořit objekt MessageProducer pro místo určení. Chcete-li odeslat zprávu do místa určení, aplikace vytvoří objekt Message a pak volá metodu send () objektu MessageProducer .

Aplikace používá objekt MessageProducer k odesílání zpráv. Aplikace obvykle vytvoří objekt MessageProducer pro určité místo určení, což může být fronta nebo téma, takže všechny zprávy odeslané s producentem zpráv budou odeslány do stejného cíle. Proto, než aplikace může vytvořit objekt MessageProducer , musí nejprve vytvořit objekt Fronta nebo Téma. Informace o tom, jak vytvořit objekt Fronta nebo Téma, najdete v následujících tématech:

- “Použití JNDI k načtení spravovaných objektů v aplikaci JMS” na stránce 836
- “Použití rozšíření IBM JMS” na stránce 837
- “Použití rozšíření WebSphere MQ JMS” na stránce 843
- “Vytvoření cílů v aplikaci JMS” na stránce 848

Chcete-li vytvořit objekt MessageProducer , aplikace použije metodu createProducer() objektu relace, jak je uvedeno v následujícím příkladu:

```
MessageProducer producer = session.createProducer(destination);
```

Parametr `destination` je objekt Queue nebo Topic, který aplikace vytvořila dříve.

Dříve než může aplikace odeslat zprávu, musí vytvořit objekt Zpráva. Tělo zprávy obsahuje data aplikace a služba JMS definuje pět typů těla zprávy:

- Bajtů
- Mapa
- Objekt
- Proud
- Text

Každý typ těla zprávy má své vlastní rozhraní JMS, což je podrozhraní rozhraní zpráv a metoda v rozhraní relace pro vytvoření zprávy s daným typem těla. Například rozhraní pro textovou zprávu se nazývá TextMessagea aplikace používá metodu createTextMessage () objektu relace k vytvoření textové zprávy, jak je uvedeno v následujícím příkazu:

```
TextMessage outMessage = session.createTextMessage(outString);
```

Další informace o zprávách a tělech zpráv najdete v tématu “Zprávy JMS” na stránce 780.

Chcete-li odeslat zprávu, aplikace použije metodu send () objektu MessageProducer , jak je zobrazeno v následujícím příkladu:

```
producer.send(outMessage);
```

Aplikace může používat metodu send () k odesílání zpráv v doméně systému zpráv. Povaha místa určení určuje, která doména systému zpráv se používá. Avšak, TopicPublisher, podřízené rozhraní MessageProducer , který je specifický pro doménu publikování/odběru, má také metodu publish (), kterou lze použít místo metody send (). Tyto dvě metody jsou funkčně stejné.

Aplikace může vytvořit objekt MessageProducer , který nemá žádné zadané místo určení. V takovém případě musí aplikace určit místo určení při volání metody send ().

Pokud aplikace odešle zprávu v rámci transakce, zpráva nebude doručena do místa určení, dokud nebude transakce potvrzena. To znamená, že aplikace nemůže odeslat zprávu a přijmout odpověď na zprávu v rámci stejné transakce.

Místo určení lze konfigurovat tak, že když aplikace odešle zprávy do této zprávy, třídy WebSphere MQ pro platformu JMS předá zprávu a vrátí řízení zpět aplikaci, aniž by určila, zda správce front zprávu obdržel

bezpečně. Toto je někdy označováno jako *asynchronní vložení*. Další informace viz “Asynchronně vkládání zpráv do tříd produktu IBM WebSphere MQ pro platformu JMS” na stránce 889.

Příjem zpráv v aplikaci JMS

Aplikace používá pro příjem zpráv spotřebitele zpráv. Odběratel trvalého tématu je spotřebitel zpráv, který přijímá všechny zprávy odeslané do místa určení, včetně těch, které byly odeslány, zatímco odběratel je neaktivní. Aplikace může vybrat zprávy, které chce přijímat, pomocí selektoru zpráv, a může přijímat zprávy asynchronně pomocí modulu listener pro zprávy.

Aplikace používá objekt MessageConsumer k příjmu zpráv. Aplikace vytvoří objekt MessageConsumer pro specifické místo určení, což může být fronta nebo téma, takže všechny zprávy přijaté s použitím spotřebitele zpráv budou přijaty ze stejného cíle. Dříve než aplikace může vytvořit objekt MessageConsumer , musí nejprve vytvořit objekt Fronta nebo Téma. Informace o tom, jak vytvořit objekt Fronta nebo Téma, najdete v následujících tématech:

- “Použití JNDI k načtení spravovaných objektů v aplikaci JMS” na stránce 836
- “Použití rozšíření IBM JMS” na stránce 837
- “Použití rozšíření WebSphere MQ JMS” na stránce 843
- “Vytvoření cílů v aplikaci JMS” na stránce 848

Chcete-li vytvořit objekt MessageConsumer , aplikace použije metodu createConsumer() objektu relace, jak je uvedeno v následujícím příkladu:

```
MessageConsumer consumer = session.createConsumer(destination);
```

Parametr `destination` je objekt Queue nebo Topic, který aplikace vytvořila dříve.

Aplikace potom použije metodu receive () objektu MessageConsumer k přijetí zprávy z místa určení, jak je zobrazeno v následujícím příkladu:

```
Message inMessage = consumer.receive(1000);
```

Parametr ve volání metody receive () uvádí, jak dlouho v milisekundách metoda čeká na příchod vyhovující zprávy, pokud není okamžitě k dispozici žádná zpráva. Vynecháte-li tento parametr, budou bloky volání po dobu neurčitou do té doby, než dorazí vhodná zpráva. Nechcete-li, aby aplikace čekala na zprávu, použijte místo toho metodu wait () receiveNoWait ().

Metoda receive () vrací zprávu určitého typu. Když například aplikace obdrží textovou zprávu, objekt vrácený voláním receive () je objekt TextMessage .

Avšak deklarovaný typ objektu vrácený voláním receive () je objekt zprávy. Proto, aby bylo možné extrahovat data z těla zprávy, která byla právě přijata, musí aplikace přetypovat ze třídy Message na specifičtější podtřídu, jako např. TextMessage. Není-li typ zprávy znám, může aplikace použít operátor `instanceof` k určení typu zprávy. Pro aplikaci je vždy dobrým zvykem určit typ zprávy před odléváním, aby bylo možné s chybami zacházet elegantně.

Následující kód používá operátor `instanceof` a ukazuje, jak extrahovat data z těla textové zprávy:

```
if (inMessage instanceof TextMessage) {
  String replyString = ((TextMessage) inMessage).getText();
  .
  .
  .
} else {
  // Print error message if Message was not a TextMessage.
  System.out.println("Reply message was not a TextMessage");
}
```

Pokud aplikace odešle zprávu v rámci transakce, zpráva nebude doručena do místa určení, dokud nebude transakce potvrzena. To znamená, že aplikace nemůže odeslat zprávu a přijmout odpověď na zprávu v rámci stejné transakce.

Pokud spotřebitel zpráv přijímá zprávy z místa určení, které je konfigurováno pro dopředné čtení, všechny přechodné zprávy, které jsou ve vyrovnávací paměti čtení napřed při ukončení aplikace, budou vyřazeny.

V doméně publikování/odběru služba JMS identifikuje dva typy odběratelů zpráv, přechodného odběratele tématu a trvalého odběratele tématu, které jsou popsány v následujících dvou sekcích.

Odběratelé tématu Nondurable

Odběratel netrvalého tématu přijímá pouze zprávy, které byly publikovány v době, kdy je odběratel aktivní. Netrvalý odběr se spustí, když aplikace vytvoří netrvalého odběratele tématu a končí, když aplikace zavře odběratele, nebo když odběratel přestane mít rozsah. Jako rozšíření v rámci tříd WebSphere MQ pro platformu JMS také odběratel netrvalého tématu přijímá i zachované publikace, ale nikoli při použití připojení v reálném čase ke zprostředkovateli.

Chcete-li vytvořit netrvalého odběratele tématu, může aplikace použít metodu createConsumer() nezávislou na doméně a určit jako cíl objekt Topic. Alternativně může aplikace použít metodu specifickou pro doménu createSubscriber(), jak je uvedeno v následujícím příkladu:

```
TopicSubscriber subscriber = session.createSubscriber(topic);
```

Parametr `topic` je objekt Topic, který aplikace vytvořila dříve.

Odběratelé trvalého tématu

Omezení: Aplikace nemůže vytvořit trvalé odběratele tématu při použití připojení v reálném čase ke zprostředkovateli.

Odběratel trvalého tématu přijímá všechny zprávy, které byly publikovány během životnosti trvalého odběru. Tyto zprávy zahrnují všechny ty zprávy, které jsou publikovány v době, kdy odběratel není aktivní. Odběratel trvalého tématu jako rozšíření v produktu WebSphere MQ pro platformu JMS také přijímá zachované publikace.

Chcete-li vytvořit odběratele trvalého tématu, aplikace použije metodu createDurableSubscriber () objektu Session, jak ukazuje následující příklad:

```
TopicSubscriber subscriber = session.createDurableSubscriber(topic, "D_SUB_000001");
```

Na volání createDurableSubscriber () je první parametr objekt Topic, který aplikace vytvořila dříve, a druhý parametr je název, který se používá k identifikaci trvalého odběru.

Relace použitá k vytvoření odběratele trvalého tématu musí mít přidružený identifikátor klienta. Identifikátor klienta přidružený k relaci je stejný jako identifikátor klienta pro připojení, které se používá k vytvoření relace. Identifikátor klienta lze zadat tak, že nastavíte vlastnost CLIENTID objektu ConnectionFactory . Alternativně může aplikace uvést identifikátor klienta voláním metody ID () ID objektu setClientobjektu Connection.

Název použitý k identifikaci trvalého odběru musí být jedinečný pouze v rámci identifikátoru klienta, a proto je identifikátor klienta součástí úplného, jedinečného identifikátoru trvalého odběru. Chcete-li nadále používat trvalý odběr, který byl vytvořen dříve, musí aplikace vytvořit trvalého odběratele témat pomocí relace se stejným identifikátorem klienta, jaký je přidružen k trvalému odběru, a s použitím stejného názvu odběru.

Trvalý odběr se spustí, když aplikace vytvoří trvalého odběratele tématu pomocí identifikátoru klienta a názvu odběru, pro který momentálně neexistuje trvalý odběr. Trvalý odběr však není ukončen, pokud aplikace zavře odběratele trvalého tématu. Chcete-li ukončit trvalý odběr, musí aplikace volat metodu unsubscribe () objektu relace, který má stejný identifikátor klienta jako přidružený k trvalému odběru. Parametr ve volání zrušení odběru () je názvem odběru, jak je zobrazeno v následujícím příkladu:

```
session.unsubscribe("D_SUB_000001");
```

Rozsah trvání trvalého odběru je správce front. Pokud existuje trvalý odběr v jednom správci front a aplikace připojená k jinému správci front vytvoří trvalý odběr se stejným identifikátorem klienta a názvem odběru, jsou dva trvalé odběry zcela nezávislé.

Voliče zpráv

Aplikace může určit, že po sobě jdoucí volání receive () vrátí pouze zprávy, které splňují určitá kritéria. Při vytváření objektu MessageConsumer může aplikace určit výraz jazyka SQL (Structured Query Language), který určuje, které zprávy jsou načítány. Tento výraz SQL se nazývá *selektor zpráv*. Selektor zpráv může obsahovat názvy polí záhlaví zpráv JMS a vlastností zpráv. Informace o tom, jak vytvořit selektor zpráv, viz “Selektory zpráv v JMS” na stránce 781.

Následující příklad ukazuje, jak aplikace může vybírat zprávy založené na vlastnosti definované uživatelem s názvem myProp:

```
MessageConsumer consumer;
.
consumer = session.createConsumer(destination, "myProp = 'blue'");
```

Specifikace JMS nepovoluje aplikaci změnit selektor zpráv pro spotřebitele zpráv. Poté, co aplikace vytvoří spotřebitele zpráv se selektorem zpráv, zůstane selektor zpráv po celou dobu životnosti tohoto spotřebitele. Pokud aplikace vyžaduje více než jeden selektor zpráv, aplikace musí vytvořit spotřebitele zpráv pro každý selektor zpráv.

Všimněte si, že když je aplikace připojena ke správci front verze 7, nemá vlastnost MSGSELECTION továrny připojení žádný efekt. Chcete-li optimalizovat výkon, všechny volby zpráv provádí správce front.

Potlačení lokálních publikování

Aplikace může vytvořit spotřebitele zpráv, který ignoruje publikování publikované ve vlastním připojení spotřebitele. Aplikace to provede nastavením třetího parametru na volání createConsumer() na `true`, jak je uvedeno v následujícím příkladu:

```
MessageConsumer consumer = session.createConsumer(topic, null, true);
```

Na volání createDurableSubscriber () to aplikace provede nastavením čtvrtého parametru na hodnotu `true`, jak je uvedeno v následujícím příkladu.

```
String selector = "company = 'IBM'";
TopicSubscriber subscriber = session.createDurableSubscriber(topic, "D_SUB_000001",
                                                              selector, true);
```

Asynchronní doručení zpráv

Aplikace může přijímat zprávy asynchronně pomocí registrace modulu listener zpráv se spotřebitelem zpráv. Modul listener pro zprávy má metodu s názvem onMessage, která se volá asynchronně, je-li k dispozici vhodná zpráva a jejímž účelem je zpracování zprávy. Následující kód ilustruje tento mechanizmus:

```
import javax.jms.*;

public class MyClass implements MessageListener
{
  // The method that is called asynchronously when a suitable message is available
  public void onMessage(Message message)
  {
    System.out.println("Message is "+message);

    // The code to process the message
    .
    .
    .
  }
}
.
.
.
// Main program (possibly in another class)
.
// Creating the message listener
MyClass listener = new MyClass();
```

```
// Registering the message listener with a message consumer
consumer.setMessageListener(listener);

// The main program now continues with other processing
```

Aplikace může použít relaci buď pro příjem zpráv synchronně pomocí volání receive (), nebo pro asynchronní příjem zpráv pomocí listenerů zpráv, ale ne pro obojí. Musí-li aplikace přijímat zprávy synchronně a asynchronně, musí vytvořit samostatné relace.

Jakmile je relace nastavena k asynchronnímu příjmu zpráv, nemohou být na této relaci nebo na objektech vytvořených z této relace volány následující metody:

- MessageConsumer.receive ()
- MessageConsumer.receive (long)
- MessageConsumer.receiveNoWait ()
- Session.acknowledge()
- MessageProducer.send (Místo určení, Zpráva)
- MessageProducer.send (cíl, zpráva, int, int, long)
- MessageProducer.send (Message)
- MessageProducer.send (Message, int, int, long).
- Session.commit()
- Session.createBrowser(Fronta)
- Session.createBrowser(Fronta, Řetězec)
- Session.createBytesMessage()
- Session.createConsumer(Cíl)
- Session.createConsumer(Cíl, String, logická hodnota)
- Session.createDurableSubscriber(Topic, String)
- Session.createDurableSubscriber(Topic, String, String, boolean)
- Session.createMapMessage()
- Session.createMessage()
- Session.createObjectMessage()
- Session.createObjectMessage(serializovatelná)
- Session.createProducer(Cíl)
- Session.createQueue(String)
- Session.createStreamMessage()
- Session.createTemporaryQueue()
- Session.createTemporaryTopic()
- Session.createTextMessage()
- Session.createTextMessage(Řetězec)
- Session.createTopic()
- Session.getAcknowledgeMode()
- Session.getMessageListener()
- Session.getTransacted()
- Session.rollback()
- Session.unsubscribe(String)

Je-li vyvolána některá z těchto metod, bude vyvolána výjimka JMSException obsahující zprávu:

```
JMSCC0033: Volání synchronní metody není povoleno, když je relace používána asynchronně: 'method
name'
```

je vyhozena.

Příjem nezpracovatelných zpráv

Aplikace může přijmout zprávu, která nemůže být zpracována. Může existovat několik důvodů, proč zprávu nelze zpracovat, například zpráva může mít chybný formát. Takové zprávy jsou popsány jako nezpracovatelné zprávy a vyžadují speciální zacházení, aby se zabránilo rekurzivnímu zpracování zprávy.

Podrobnosti o zpracování nezpracovatelných zpráv najdete v tématu “Zpracování nezpracovatelných zpráv v produktu IBM WebSphere MQ classes for JMS” na stránce 859.

V 7.5.0.8 Načtení uživatelských dat odběru

Pokud zprávy, které aplikace IBM WebSphere MQ classes for JMS spotřebovávají z fronty, jsou z administrativně definovaného trvalého odběru uvedeny administrativně, musí aplikace přistupovat k informacím o uživateli, které jsou k odběru přidruženy. Tyto informace se přidají do zprávy jako vlastnost.

Pokud je v produktu Version 7.5.0, Fix Pack 8zpráva spotřebována z fronty obsahující záhlaví RFH2 se složkou MQPS, je hodnota přidružená ke klíči Sud, pokud existuje, přidána jako vlastnost řetězce do objektu zprávy JMS vráceného do aplikace produktu IBM WebSphere MQ classes for JMS . Chcete-li povolit načtení této vlastnosti ze zprávy, lze použít konstantu JMS_IBM_SUBSCRIPTION_USER_DATA v rozhraní JmsConstants s metodou javax.jms.Message.getStringProperty(java.lang.String) k získání uživatelských dat odběru.

V následujícím příkladu je definován administrativní trvalý odběr s použitím příkazu MQSC **DEFINE SUB**:

```
DEFINE SUB('MY.SUBCRIPTION') TOPICSTR('PUBLIC') DEST('MY.SUBSCRIPTION.Q')
USERDATA('Administrative durable subscription to put message to the queue MY.SUBSCRIPTION.Q')
```

Kopie zpráv, které jsou publikovány do řetězce tématu PUBLIC , jsou vloženy do fronty MY.SUBSCRIPTION.Q. Uživatelská data, která jsou přidružená k trvalému odběru, jsou poté přidána jako vlastnost do zprávy, která je uložena ve složce MQPS záhlaví RFH2 s klíčem Sud.

Aplikace IBM WebSphere MQ classes for JMS může volat:

```
javax.jms.Message.getStringProperty(JmsConstants.JMS_IBM_SUBSCRIPTION_USER_DATA);
```

Pak se vrátí následující řetězec:

```
Administrative durable subscription to put message to the queue MY.SUBSCRIPTION.Q
```

.

Související pojmy

“Záhlaví a rozhraní JMS MQRFH2 .” na stránce 785

Související úlohy

Definování administrativního odběru

Související odkazy

DEFINE SUB

Rozhraní JmsConstants

Zavření tříd produktu WebSphere MQ pro aplikaci JMS

Je důležité, aby třídy produktu WebSphere MQ pro aplikaci JMS explicitně zavřela určité objekty JMS před zastavením. Finalizéry nemusí být volány, takže se na ně nespoléhejte na volné prostředky. Nepovolujte ukončení aplikace s aktivním trasovaným trasováním.

Samotné uvolňování paměti nemůže včas uvolnit všechny třídy WebSphere MQ pro prostředky JMS a WebSphere MQ , zvláště pokud aplikace vytváří mnoho krátkých objektů JMS s krátkou životností na úrovni relace nebo nižší. Je proto důležité, aby aplikace zavřela objekt Connection, Session, MessageConsumernebo objekt MessageProducer , když již není potřeba.

Pokud aplikace skončí bez zavření připojení, dojde k implicitnímu odvolání transakce pro všechny relace obsažené v relaci s transakcemi. Chcete-li zajistit, aby byly provedeny změny provedené aplikací, před zavřením aplikace uzavřete spojení explicitně.

V aplikaci nepoužívejte finalizéry k uzavření objektů JMS. Protože nelze moduly finalizer volat, prostředky nemusí být uvolněny. Když je spojení uzavřeno, zavře všechny relace, které z ní byly vytvořeny. Podobně platí, že MessageConsumers a MessageProducers vytvořené z relace jsou zavřeny při zavření relace. Nicméně, zvažte uzavření relací, MessageConsumersa MessageProducers explicitně k zajištění toho, že prostředky budou uvolněny včas.

Je-li aktivována komprese trasování, System.Halt() vypnutí a abnormální, nekontrolované ukončení prostředí JVM pravděpodobně povede k poškození trasovacího souboru. Je-li to možné, vypněte trasovací prostředek, pokud jste shromáždili trasovací informace, které potřebujete. Pokud sledujete aplikaci až do abnormálního ukončení, použijte nekomprimovaný trasovací výstup.

Zpracování nezpracovatelných zpráv v produktu IBM WebSphere MQ classes for JMS

Pozastavené zprávy jsou takové, které nemohou být zpracovány přijímající aplikací MDB. Je-li zjištěna nezpracovatelná zpráva, objekty JMS MessageConsumer a ConnectionConsumer ji mohou znovu zařadit podle dvou vlastností fronty, BOQNAME a BOTHRESH.

Někdy je do fronty doručena špatně formátovaná zpráva. V tomto kontextu nesprávně naformátovaný znamená, že přijímající aplikace nemůže správně zpracovat zprávu. Taková zpráva může způsobit selhání přijímající aplikace a vrácení této špatně formátované zprávy. Zprávu lze poté opakovaně doručit do vstupní fronty a opakovaně ji podporovat aplikací. Tyto zprávy se označují jako *nezpracovatelné zprávy*. Objekt JMS MessageConsumer zjistí nezpracovatelné zprávy a přesměrovává je na alternativní místo určení.

Správce front produktu IBM WebSphere MQ uchovává záznamy o počtu případů, kdy byla každá zpráva vrácena zpět. Když tento počet dosáhne konfigurovatelné prahové hodnoty, spotřebitel zpráv znovu odešle zprávu do pojmenované výstupní fronty. Pokud z nějakého důvodu tento požadavek selže, zpráva bude odebrána ze vstupní fronty a buď znovu zařazena do fronty nedoručených zpráv, nebo je vyřazena. Další informace viz část “Odebrání zpráv z fronty v ASF” na stránce 897.

Existuje rozdíl mezi tím, jak MessageConsumers a ConnectionConsumersrekvalifikoval nezpracovatelné zprávy. Volba ConnectionConsumers umožňuje vrácení nezpracovatelných zpráv do fronty bez ovlivnění doručení zprávy. Proces opětovného zařazení se provádí mimo jakoukoli jednotku práce přidruženou ke skutečnému doručení zprávy do kódu aplikace. To je možné díky vícevláknové povaze operace ConnectionConsumer .

MessageConsumersjsou však jednotlivé podprocesy pod úrovní relace a všechny zprávy o škodných zprávách se nacházejí v aktuální jednotce práce. To nemá vliv na operaci aplikace, avšak při opětném zařazení nezpracovatelných zpráv do relace nebo relace klienta potvrzení se akce nepotvrzuje, dokud nebude aktuální transakce potvrzena kódem aplikace nebo případně kódem kontejneru aplikací.

Objekty JMS 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 před tím, než dojde k opětovnému zařazení zpráv do fronty. Toto chování je způsobeno tím, jak jednotliví spotřebitelé připojení monitorují fronty a přeřadí nezpracovatelné zprávy do fronty.

Prahová hodnota a název back-výstupní fronty jsou atributy fronty IBM WebSphere MQ . Názvy atributů jsou BackoutThreshold a BackoutRequeueQName. Fronta, na kterou se vztahují, je následující:

- Pro systém zpráv typu point-to-point je to základní lokální fronta. To je důležité, pokud spotřebitelé zpráv a spotřebitelé připojení používají aliasy fronty.
- Pro systém zpráv typu publikování/odběr v běžném režimu poskytovatele systému zpráv produktu IBM WebSphere MQ je fronta spravovaná prostřednictvím této fronty vytvořena z modelové fronty.

- V případě systému zpráv typu publikování/odběr v režimu migrace poskytovatele systému zpráv produktu IBM WebSphere MQ se jedná o frontu CCSUB definovanou v objektu továrny TopicConnectionnebo ve frontě CCDSUB, která je definována v objektu Topic.

Produkt IBM WebSphere MQ classes for JMS se dotazuje na název fronty BackoutThreshold a BackoutRequeueQName fronty. Proto je třeba uživateli, který aplikaci spustil, udělit přístup pro dotazy k frontě.

V 7.5.0.9 Je-li cílovou frontou fronta klastru, pak potřebné oprávnění závisí na použité verzi IBM WebSphere MQ classes for JMS :

- Když používáte IBM WebSphere MQ classes for JMS for Version 7.5.0, Fix Pack 9 plus prozatímní opravu pro APAR IT26482, je požadován přístup k dotazu.
- Pro všechny ostatní verze udělte dotazům, procházejte a získejte přístup.

Chcete-li nastavit atributy QName BackoutThreshold a BackoutRequeue, zadejte následující příkaz MQSC:

```
ALTER QLOCAL(your.queue.name) BOTHRESH(threshold value) BOQNAME(your.backout.queue.name)
```

Je-li atribut BackoutThreshold nastaven na jinou hodnotu než nula, chcete-li se vyhnout neočekávanému chování, nastavte atribut QName BackoutRequeuena platný název fronty.

Pokud váš systém pro systém zpráv typu publikování/odběr vytváří pro každý odběr dynamickou frontu, jsou tyto hodnoty atributů získány z modelové fronty IBM WebSphere MQ classes for JMS , SYSTEM.JMS.MODEL.QUEUE. Chcete-li tato nastavení změnit, použijte:

```
ALTER QMODEL(SYSTEM.JMS.MODEL.QUEUE) BOTHRESH(threshold value) BOQNAME(your.backout.queue.name)
```

Je-li prahová hodnota vrácení nula, zpracování nezpracovatelných zpráv je zakázáno a nezpracovatelné zprávy zůstanou ve vstupní frontě. Jinak, když se počet vrácení dosáhne prahové hodnoty, zpráva se odešle do pojmenované fronty vrácení. Pokud počet vrácení dosáhne prahové hodnoty, ale zpráva nemůže jít do fronty vyřazených zpráv, odešle se zpráva do fronty nedoručených zpráv nebo je vyřazena. Tato situace nastane, pokud není definována fronta vrácení, nebo pokud objekt MessageConsumer nemůže odeslat zprávu do fronty vrácení. Další podrobnosti najdete v tématu “Odebrání zpráv z fronty v ASF” na stránce 897 .

Je-li zpráva znovu zařazena do fronty vrácených zpráv, některé z hodnot polí v deskriptoru zpráv MQMD (Message Descriptor) se změní. Podrobnosti o formátu MQMD viz MQMD-Message Descriptor .

Následující hodnota pole MQMD se změní, když se zpráva dostane do fronty vrácení.

- Hodnota PutDate je aktualizována na datum, do kterého patří do fronty vrácených zpráv.
- Funkce PutTime se aktualizuje na čas, kdy putuje do fronty vrácených zpráv.
- Počet vrácení je resetován na nulu.
- Vypršení platnosti zprávy je aktualizováno tak, aby odráželo zbývající dobu platnosti v době, kdy byla aplikací JMS přijata původní zpráva.

Hodnoty v následujících polích zůstanou stejné, když se zpráva dostane do fronty vyřazovacích zpráv:

- StructId
- Verze
- Sestava
- MessageType
- Zpětná vazba
- Kódování
- CodedCharSetId
- MsgId
- CorrelId
- ReplyToQ

- ReplyToQMgr
- Formát
- Trvání
- Priorita

Výjimky v IBM WebSphere MQ classes for JMS

Aplikace produktu IBM WebSphere MQ classes for JMS musí být schopna zpracovávat výjimky vyvolané voláním rozhraní API JMS nebo doručené do obslužné rutiny výjimek.

Produkt IBM WebSphere MQ classes for JMS vykazuje běhové problémy vyvoláním výjimek. JMSException je kořenová třída pro výjimky vyvolané metodami JMS a zachycení výjimek JMSException poskytuje generický způsob obsluhy všech výjimek souvisejících s platformou JMS.

Každá výjimka JMSException zapouzdřuje následující informace:

- Zpráva o výjimce specifické pro poskytovatele, kterou aplikace získá voláním metody Throwable.getMessage().
- Kód chyby specifický pro poskytovatele, který aplikace získá voláním metody JMSException.getErrorCode().
- Propojená výjimka. Výjimka vyvolaná voláním rozhraní JMS API je často výsledkem problému nižší úrovně, který je hlášen jinou výjimkou spojenou s touto výjimkou. Aplikace získá spojenou výjimku voláním metody JMSException.getLinkedException() nebo metody Throwable.getCause().

Většina výjimek vygenerovaných produktem IBM WebSphere MQ classes for JMS jsou instance podtříd výjimky JMSException. Tyto podtřídy implementují rozhraní com.ibm.msg.client.jms.JmsExceptionDetail , které poskytuje následující dodatečné informace:

- Vysvětlení zprávy o výjimce, kterou aplikace získá voláním metody JmsExceptionDetail.getExplanation().
- Doporučená odezva uživatele na výjimku, kterou aplikace získá voláním metody JmsExceptionDetail.getUserAction().
- Klíče pro vložení zpráv ve zprávě výjimky. Aplikace získá iterátor pro všechny klíče pomocí volání metody JmsExceptionDetail.getKeys().
- Zpráva se vloží do zprávy výjimky. Vložením zprávy může být například název fronty, která výjimku způsobila, a může být užitečné, aby aplikace mohla přistupovat k tomuto názvu. Aplikace získá vložení zprávy odpovídající uvedenému klíči voláním metody JmsExceptionDetail.getValue().

Všechny metody v rozhraní Podrobnosti JmsExceptionmohou vracet hodnotu null, nejsou-li k dispozici žádné podrobnosti.

Pokud se například aplikace pokusí vytvořit producenta zpráv pro frontu IBM WebSphere MQ , která neexistuje, je vyvolána výjimka s následujícími informacemi:

```
Message : JMSWMQ2008: Failed to open MQ queue 'Q_test'.
Class : class com.ibm.msg.client.jms.DetailedInvalidDestinationException
Error Code : JMSWMQ2008
Explanation : JMS attempted to perform an MQOPEN, but WebSphere MQ reported an
              error.
User Action : Use the linked exception to determine the cause of this error. Check
              that the specified queue and queue manager are defined correctly.
```

Vyvolaná výjimka com.ibm.msg.client.jms.DetailedInvalidDestinationExceptionje podtřídou třídy javax.jms.InvalidDestinationException a implementuje rozhraní com.ibm.msg.client.jms.JmsExceptionDetail .

Propojené výjimky

Propojená výjimka poskytuje další informace o běhovém problému. Proto pro každou výjimku JMSException, která je vyvolána, by měla aplikace zkontrolovat připojenou výjimku. Samotná připojená výjimka může mít jinou propojené výjimky a propojené výjimky tvoří řetězec vedoucí zpět k původnímu základnímu problému. Propojená výjimka je implementována použitím mechanizmu výjimek

v řetězové výjimce třídy java.lang.Throwable a aplikace obdrží připojenou výjimku voláním metody Throwable.getCause(). Pro výjimku JMSException metoda getLinkedException () ve skutečnosti deleguje metodu Throwable.getCause().

Pokud například aplikace určuje nesprávné číslo portu při připojování ke správci front, budou tyto výjimky tvořit následující řetězec:

```
com.ibm.msg.client.jms.DetailIllegalStateException
   |
   +--->com.ibm.mq.MQException
           |
           +--->com.ibm.mq.jmqi.JmqiException
                   |
                   +--->java.net.ConnectionException
```

Obvykle je každá výjimka v řetězci vyvolána z jiné vrstvy v kódu. Například výjimky v předchozím řetězci jsou vyvoláženy následujícími vrstvami:

- První výjimka, instance podtřídy JMSException, je vyvolána společnou vrstvou v produktu IBM WebSphere MQ classes for JMS.
- Další výjimka, instance com.ibm.mq.MQException, je vyvolána poskytovatelem systému zpráv produktu IBM WebSphere MQ .
- Další výjimka: instance com.ibm.mq.jmqi.JmqiException, je generována společným rozhraním Java pro rozhraní MQI.
- Poslední výjimka, instance java.net.ConnectionException, je vyvolána knihovnou třídy Java.

Další informace o vrstvené architektuře produktu IBM WebSphere MQ classes for JMSviz “Třídy IBM WebSphere MQ pro architekturu JMS” na stránce 771.

Pomocí kódu podobného následujícímu kódu může aplikace iterovat přes tento řetězec a extrahovat všechny příslušné informace:

```
import com.ibm.msg.client.jms.JmsExceptionDetail;
import com.ibm.mq.MQException;
import com.ibm.mq.jmqi.JmqiException;
import javax.jms.JMSException;
.
.
.
catch (JMSException je) {
    System.err.println("Caught JMSException");

    // Check for linked exceptions in JMSException
    Throwable t = je;
    while (t != null) {
        // Write out the message that is applicable to all exceptions
        System.err.println("Exception Msg: " + t.getMessage());
        // Write out the exception stack trace
        t.printStackTrace(System.err);

        // Add on specific information depending on the type of exception
        if (t instanceof JMSException) {
            JMSException je1 = (JMSException) t;
            System.err.println("JMS Error code: " + je1.getErrorCode());

            if (t instanceof JmsExceptionDetail){
                JmsExceptionDetail jed = (JmsExceptionDetail)je1;
                System.err.println("JMS Explanation: " + jed.getExplanation());
                System.err.println("JMS Explanation: " + jed.getUserAction());
            }
        } else if (t instanceof MQException) {
            MQException mqe = (MQException) t;
            System.err.println("WMQ Completion code: " + mqe.getCompCode());
            System.err.println("WMQ Reason code: " + mqe.getReason());
        } else if (t instanceof JmqiException){
            JmqiException jmqie = (JmqiException)t;
            System.err.println("WMQ Log Message: " + jmqie.getWmqLogMessage());
            System.err.println("WMQ Explanation: " + jmqie.getWmqMsgExplanation());
            System.err.println("WMQ Msg Summary: " + jmqie.getWmqMsgSummary());
            System.err.println("WMQ Msg User Response: "
                               + jmqie.getWmqMsgUserResponse());
            System.err.println("WMQ Msg Severity: " + jmqie.getWmqMsgSeverity());
        }
```

```
        // Get the next cause
        t = t.getCause();
    }
}
```

Vezměte na vědomí, že aplikace by měla vždy kontrolovat typ každé výjimky v řetězu, protože typ výjimky se může lišit a výjimky různých typů zapouzdřují různé informace.

Získání specifických informací IBM WebSphere MQ o problému

Instance com.ibm.mq.MQException a com.ibm.mq.jmqi.JmqiException zapouzdřují IBM WebSphere MQ specifické informace o problému.

Výjimka MQException zapouzdřuje následující informace:

- Kód dokončení, který aplikace získá voláním metody Code () getComp
- Kód příčiny, který aplikace získá voláním metody getReason()

Výjimka JmqiException také zapouzdřuje kód dokončení a kód příčiny. Kromě toho však výjimka JmqiException zapouzdřuje informace ve zprávě AMQ*nnnn* nebo CSQ*nnnn* , je-li k výjimce přidružena výjimka. Voláním odpovídajících metod této výjimky může aplikace získat různé komponenty této zprávy, jako je závažnost, vysvětlení a odezva uživatele.

Příklady použití metod uvedených v tomto oddílu naleznete v ukázkovém kódu v produktu “Propojené výjimky” na stránce 861.

Upgrade z předchozích verzí produktu IBM WebSphere MQ classes for JMS

Ve srovnání s předchozími verzemi produktu IBM WebSphere MQ classes for JMSse většina chybových kódů a zpráv výjimek změnila ve verzi 7. Důvodem těchto změn je, že produkt IBM WebSphere MQ classes for JMS má nyní vrstvenou architekturu a výjimky jsou vyvoláženy různými vrstvami v kódu.

Pokud se například aplikace pokouší o připojení ke správci front, který neexistuje, předchozí verze produktu IBM WebSphere MQ classes for JMS vygenerovala výjimku JMSException s následujícími informacemi:

```
MQJMS2005: Failed to create MQQueueManager for 'localhost:QM_test'.
```

Tato výjimka obsahovala výjimku propojené výjimky MQException s následujícími informacemi:

```
MQJE001: Completion Code 2, Reason 2058
```

Ve srovnání se stejnými okolnostmi vrací verze 7 produktu IBM WebSphere MQ classes for JMS výjimku JMSException s následujícími informacemi:

```
Message : JMSWMQ0018: Failed to connect to queue manager 'QM_test' with
          connection mode 'Client' and host name 'localhost'.
Class : class com.ibm.msg.client.jms.DetailedJMSException
Error Code : JMSWMQ0018
Explanation : null
User Action : Check the queue manager is started and if running in client mode,
              check there is a listener running. Please see the linked exception
              for more information.
```

Tato výjimka obsahuje spojenou výjimku MQException s následujícími informacemi:

```
Message : JMSCMQ0001: WebSphere MQ call failed with compcode '2' ('MQCC_FAILED')
          reason '2058' ('MQRC_Q_MGR_NAME_ERROR').
Class : class com.ibm.mq.MQException
Completion Code : 2
Reason Code : 2058
```

Pokud vaše aplikace analyzuje nebo testuje výjimky vrácené metodou Throwable.getMessage() nebo chybové kódy vrácené metodou JMSException.getErrorCode() a provádíte upgrade z vydání před verzí 7,

je třeba aplikaci pravděpodobně upravit, aby používala verzi 7 produktu IBM WebSphere MQ classes for JMS.

Moduly listener výjimek

Aplikace může registrovat modul listener pro výjimky s objektem připojení. Pokud se následně vyskytne problém, který znemožňuje připojení, produkt IBM WebSphere MQ classes for JMS doručí výjimku do modulu listener výjimek vyvoláním metody onException(). Aplikace pak má možnost znovu ustanovit spojení.

V 7.5.0.8 APAR IT14820, zahrnutý z produktu IBM WebSphere MQ Version 7.5.0, opravná sada Fix Pack 8, opravený defekt, kdy by rozhraní JMS ExceptionListener aplikace nebylo vyvoláno pro nesouvisející přerušené výjimky (například MQRC_GET_INHIBITED), přestože vlastnost ASYNC_EXCEPTIONS v aplikaci JMS Connection Factory použitá aplikací byla nastavena na hodnotu ASYNC_EXCEPTIONS_ALL. Toto byla předvolená hodnota před Version 7.5.0, Fix Pack 8.

V 7.5.0.8 Chcete-li zachovat chování pro aktuální aplikace JMS, které konfigurují rozhraní JMS MessageListener a JMS ExceptionListener, a aby se zajistilo, že je IBM WebSphere MQ classes for JMS konzistentní se specifikací JMS, byla výchozí hodnota pro vlastnost ASYNC_EXCEPTIONS JMS ConnectionFactory změněna na ASYNC_EXCEPTIONS_CONNECTIONBROKEN pro IBM WebSphere MQ classes for JMS. V důsledku toho jsou do modulu JMS ExceptionListener aplikace standardně doručeny pouze výjimky související s kódy chyby pro přerušené připojení.

V 7.5.0.8 From Version 7.5.0, Fix Pack 8, the IBM WebSphere MQ classes for JMS have also been updated such that JMSExceptions relating to non-connection broken errors, which occur during message delivery to asynchronous message consumers, are still delivered to a registered ExceptionListener when the JMS ConnectionFactory used by the application has the ASYNC_EXCEPTIONS property set to the value ASYNC_EXCEPTIONS_ALL.

V 7.5.0.8 Další informace o tom, co se změnilo pro moduly listener pro výjimky produktu Version 7.5.0, Fix Pack 8 a proč byly provedeny změny z předchozích verzí, najdete v tématu JMS: Modul listener pro výjimky ve verzi 7.5.

U všech ostatních typů problémů je vyvolána výjimka JMSException aktuální voláním rozhraní JMS API.

Pokud aplikace neregistruje modul listener pro výjimky s objektem připojení, všechny výjimky, které by byly doručeny do listeneru výjimek, jsou zapsány do protokolu IBM WebSphere MQ classes for JMS .

Související odkazy

VÝJIMKA ASYNCEXCEPTION

Protokolování chyb ve třídách produktu WebSphere MQ pro službu JMS

Informace o běhových problémech, které mohou vyžadovat nápravnou akci uživatele, jsou zapsány do tříd produktu WebSphere MQ pro protokol JMS.

Pokud se například aplikace pokusí nastavit vlastnost továrny připojení, ale název vlastnosti není rozpoznán, třídy WebSphere MQ pro rozhraní JMS zapíší informace o problému do svého protokolu.

Soubor obsahující protokol se standardně jmenuje mqjms.log a nachází se v aktuálním pracovním adresáři. Název a umístění souboru protokolu však můžete změnit nastavením vlastnosti com.ibm.msg.client.commonservices.log.outputName v konfiguračním souboru WebSphere MQ pro soubor konfigurace JMS. Informace o konfiguračním souboru WebSphere MQ pro konfigurační soubor JMS viz “Konfigurační soubor IBM WebSphere MQ classes for JMS” na stránce 703a další podrobnosti o platných hodnotách pro vlastnost com.ibm.msg.client.commonservices.log.outputName viz “Protokolování a IBM WebSphere MQ classes for JMS” na stránce 769.

Technologie podpory prvního selhání (FFST) v třídách WebSphere MQ pro JMS

Dojde-li k závažné interní chybě v rámci tříd WebSphere MQ pro rozhraní JMS, dojde k vygenerování informací o technologii podpory prvního selhání (FFST).

Informace o souboru FFST se zapíší do souboru s názvem JMSC*nnnn*.FDC, kde *nnnn* je čtyřmístné číslo. Tento soubor se nachází v adresáři s názvem FFDC, což je podadresář adresáře, do něhož je zapisován výstup trasování. Při výchozím nastavení je výstup trasování zapsán do aktuálního pracovního adresáře, ale můžete přesměrovat výstup trasování do jiného adresáře nastavením vlastnosti **com.ibm.msg.client.commonservices.trace.outputName** v konfiguračním souboru WebSphere MQ pro konfigurační soubor JMS. Informace o konfiguračním souboru WebSphere MQ pro konfigurační soubor JMS naleznete v tématu "Konfigurační soubor IBM WebSphere MQ classes for JMS" na stránce 703.

Je-li trasování povoleno při generování informací o produktu FFST , informace o souboru FFST se také zapíší do trasovacího souboru. Další informace o trasování programů JMS naleznete v tématu Trasování aplikací produktu IBM WebSphere MQ classes for JMS.

Chcete-li potlačit produkci souborů FFDC, nastavte vlastnost **com.ibm.msg.client.commonservices.ffst.suppress**takto:

0
Výstup všech souborů FFDC (výchozí).

-1
Výstup pouze prvních souborů FFDC určitého typu.

celočíselná hodnota
Potlačit všechny soubory FFDC kromě těch, které jsou násobkem tohoto čísla.

Přístup k funkcím produktu WebSphere MQ z tříd produktu WebSphere MQ pro aplikaci JMS

Třídy produktu WebSphere MQ pro systém JMS poskytují funkce pro využití řady funkcí produktu WebSphere MQ.

Upozornění: Tyto funkce jsou mimo specifikaci platformy JMS nebo, v určitých případech, poruší specifikaci JMS. Pokud je použijete, je nepravděpodobné, že by vaše aplikace byla kompatibilní s ostatními poskytovateli platformy JMS. Tyto funkce, které nejsou v souladu se specifikací JMS, jsou označeny upozorněním Upozornění.

Čtení a zápis deskriptoru zpráv ze tříd produktu WebSphere MQ pro aplikaci JMS

Možnost přístupu k deskriptoru zpráv (MQMD) lze řídit nastavením vlastností na cíli a ve zprávě.

Některé aplikace produktu WebSphere MQ vyžadují nastavení specifických hodnot v rámci zpráv MQMD, které jim byly odeslány. Třídy WebSphere MQ pro rozhraní JMS poskytují atributy zpráv, které umožňují aplikacím rozhraní JMS nastavit pole MQMD a umožnit tak aplikacím rozhraní JMS "řídit" aplikace WebSphere MQ .

Musíte nastavit vlastnost cílového objektu WMQ_MQMD_WRITE_ENABLED na hodnotu true pro nastavení vlastností MQMD, které mají mít nějaký vliv. Poté můžete použít metody nastavení vlastností pro zprávu (například setStringProperty) k přiřazení hodnot k polím MQMD. Všechna pole MQMD jsou odkryta s výjimkou StrucId a Verze; BackoutCount lze číst, ale ne zapisovat do.

Tento příklad vede k vložení zprávy do fronty nebo tématu s MQMD.UserIdentifier nastaven na "JoeBloggs".

```
// Create a ConnectionFactory, connection, session, producer, message
// ...

// Create a destination
// ...

// Enable MQMD write
dest.setBooleanProperty(WMQConstants.WMQ_MQMD_WRITE_ENABLED, true);

// Optionally, set a message context if applicable for this MD field
dest.setIntProperty(WMQConstants.WMQ_MQMD_MESSAGE_CONTEXT,
  WMQConstants.WMQ_MDCTX_SET_IDENTITY_CONTEXT);

// On the message, set property to provide custom UserId
msg.setStringProperty("JMS_IBM_MQMD_UserIdentifier", "JoeBloggs");
```

```
// Send the message
// ...
```

Před nastavením parametru JMS_IBM_MQMD_UserIdentifierje třeba nastavit WMQ_MQMD_MESSAGE_CONTEXT. Další informace o použití WMQ_MQMD_MESSAGE_CONTEXT viz “Vlastnosti objektu zprávy platformy JMS” na stránce 868.

Podobně můžete extrahovat obsah polí MQMD nastavením hodnoty WMQ_MQMD_READ_ENABLED na hodnotu true před přijetím zprávy a následným použitím metod get zprávy, jako je například vlastnost getString. Všechny přijaté vlastnosti jsou jen pro čtení.

Tento příklad má za výsledek pole *hodnota* , které drží hodnotu MQMD.ApplIdentityData pole zprávy bylo získáno z fronty nebo tématu.

```
// Create a ConnectionFactory, connection, session, consumer
// ...

// Create a destination
// ...

// Enable MQMD read
dest.setBooleanProperty(WMQConstants.WMQ_MQMD_READ_ENABLED, true);

// Receive a message
// ...

// Get MQMD field value using a property
String value = rcvMsg.getStringProperty("JMS_IBM_MQMD_ApplIdentityData");
```

Vlastnosti objektu místa určení platformy JMS
Dvě vlastnosti objektu Cílový objekt řídí přístup k deskriptoru MQMD z platformy JMS a třetí kontext zprávy řídí.

Tabulka 124. Názvy a popisy vlastností

Vlastnost	**Krátký formát**	**Popis**
WMQ_MQMD_WRITE_ENABLED	MDW	Určuje, zda může aplikace platformy JMS nastavit hodnoty polí MQMD
WMQ_MQMD_READ_ENABLED	MDR	Určuje, zda může aplikace platformy JMS extrahovat hodnoty polí MQMD
KONTEXT WMQ_MQMD_MESSAGE_	MDCTX	Jaká úroveň kontextu zprávy má být nastavena aplikací JMS. Aby tato vlastnost mohla nabýt účinku, musí být aplikace spuštěna s příslušným kontextovým oprávněním.

Tabulka 125. Názvy vlastností, hodnoty a metody nastavení

Vlastnost	Platné hodnoty v nástroji pro administraci (výchozí nastavení tučně)	Platné hodnoty v programech	Nastavit metodu
WMQ_MQMD_WRITE_ENABLED	• **Ne** Všechny vlastnosti JMS_IBM_MQMD* jsou ignorovány a jejich hodnoty se nezkopírují do základní struktury MQMD. • YES Vlastnosti JMS_IBM_MQMD* jsou zpracovány. Jejich hodnoty jsou zkopírovány do podkladové struktury MQMD.	• **Nepravda** • Pravda	setMQMDWritePovoleno
WMQ_MQMD_READ_ENABLED	• **Ne** Při odesílání zpráv nejsou vlastnosti JMS_IBM_MQMD* v odeslané zprávě aktualizovány tak, aby odrážely aktualizované hodnoty polí v produktu MQMD. Při příjmu zpráv nejsou dostupné žádné z vlastností JMS_IBM_MQMD* v přijaté zprávě, i když odesilatel některé či všechny tyto vlastnosti nastavil. • YES Při odesílání zpráv jsou všechny vlastnosti JMS_IBM_MQMD* v odeslané zprávě aktualizovány tak, aby odrážely aktualizované hodnoty polí v MQMD, včetně těch, které odesilatel explicitně nenastavila. Při příjmu zpráv jsou dostupné všechny vlastnosti JMS_IBM_MQMD* v přijaté zprávě včetně vlastností, které odesilatel explicitně nenastavil.	• **Nepravda** • Pravda	setMQMDReadPovoleno

Tabulka 125. Názvy vlastností, hodnoty a metody nastavení (pokračování)			
Vlastnost	**Platné hodnoty v nástroji pro administraci (výchozí nastavení tučně)**	**Platné hodnoty v programech**	**Nastavit metodu**
WMQ_MQMD _MESSAGE_CONTEXT	• **Výchozí** Volání rozhraní MQOPEN API a struktura MQPMO neurčují žádné explicitní volby kontextu zprávy. • KONTEXT SET_IDENTITY_CONTEXT Volání rozhraní MQOPEN API určuje volbu kontextu zprávy MQOO_SET_IDENTITY_CONTEXT a struktura MQPMO určuje hodnotu MQPMO_SET_IDENTITY_CONTEXT. • NASTAVITEL_VŠECH_KONTEXTU Volání rozhraní MQOPEN API uvádí volbu kontextu zprávy MQOO_SET_ALL_CONTEXT a struktura MQPMO určuje MQPMO_SET_ALL_CONTEXT	• **WMQ_MDCTX_DEFAULT** • KONTEXT WMQ_MDCTX_SET_IDENTITY_CONTEXT • WMQ_MDCTX_SET_ALL_CONTEXT	Kontext setMQMDMessage

Vlastnosti objektu zprávy platformy JMS
Vlastnosti objektu zprávy s předponou JMS_IBM_MQMD umožňují nastavit nebo přečíst odpovídající pole MQMD.

Odesílání zpráv

Všechna pole MQMD s výjimkou StrucId a verze jsou reprezentovány. Tyto vlastnosti se odkazují pouze na pole MQMD, kde se vlastnost vyskytuje jak v deskriptoru MQMD, tak v záhlaví MQRFH2 , verze v MQRFH2 není nastavena ani extrahována.

Je možné nastavit libovolnou z těchto vlastností, kromě JMS_IBM_MQMD_BackoutCount. Jakákoli hodnota nastavená pro parametr JMS_IBM_MQMD_BackoutCount je ignorována.

Má-li vlastnost maximální délku a zadáte příliš dlouhou hodnotu, bude tato hodnota zkrácena.

Pro určité vlastnosti musíte také nastavit vlastnost WMQ_MQMD_MESSAGE_CONTEXT na objektu Destination. Aby tato vlastnost mohla nabýt účinnosti, musí aplikace běžet s příslušným oprávněním kontextu. Pokud nenastavíte hodnotu WMQ_MQMD_MESSAGE_CONTEXT na příslušnou hodnotu, bude hodnota vlastnosti ignorována. Nastavíte-li WMQ_MQMD_MESSAGE_CONTEXT na příslušnou hodnotu, ale nemáte dostatečné oprávnění ke kontextu pro správce front, bude vydána výjimka JMSException. Vlastnosti vyžadující specifické hodnoty WMQ_MQMD_MESSAGE_CONTEXT jsou následující.

Následující vlastnosti vyžadují nastavení WMQ_MQMD_MESSAGE_CONTEXT na hodnotu WMQ_MDCT_X_SET_IDENTITY_CONTEXT nebo WMQ_MDCTX_SET_ALL_CONTEXT:

- JMS_IBM_MQMD_UserIdentifier .
- JMS_IBM_MQMD_AccountingToken .
- JMS_IBM_MQMD_ApplIdentityData

Následující vlastnosti vyžadují nastavení WMQ_MQMD_MESSAGE_CONTEXT na hodnotu WMQ_MDCTX_SET_ALL_CONTEXT:

- JMS_IBM_MQMD_PutApplTyp
- Název JMS_IBM_MQMD_PutAppl

- JMS_IBM_MQMD_PutDate .
- JMS_IBM_MQMD_PutTime .
- JMS_IBM_MQMD_ApplOriginData

Příjem zpráv

Všechny tyto vlastnosti jsou k dispozici v přijaté zprávě, je-li vlastnost WMQ_MQMD_READ_ENABLED nastavena na hodnotu true, bez ohledu na skutečné vlastnosti, které vytvořila aplikace pro vytváření. Aplikace nemůže upravit vlastnosti přijaté zprávy, pokud nejsou nejprve vymazány všechny vlastnosti, podle specifikace JMS. Obdrženou zprávu lze předat bez úpravy vlastností.

Upozornění: Pokud vaše aplikace přijme zprávu z cíle s vlastností WMQ_MQMD_READ_ENABLED nastaveným na hodnotu true a předá ji cíli s hodnotou WMQ_MQMD_WRITE_ENABLED nastavenou na hodnotu true, budou výsledkem všech hodnot pole MQMD přijaté zprávy do předané zprávy.

Tabulka vlastností

V této tabulce jsou uvedeny vlastnosti objektu zpráv reprezentující pole MQMD. Úplné popisy polí a jejich přípustné hodnoty najdete v odkazech.

Tabulka 126. Názvy vlastností, popisy a typy

Vlastnost	Popis	Javovský typ	Odkaz na úplný popis
JMS_IBM_MQMD_Report.	Volby pro zprávy sestav	Celé číslo	Sestava
JMS_IBM_MQMD_MsgType	Typ zprávy	Celé číslo	MsgType
JMS_IBM_MQMD_Expiry	Životnost zprávy	Celé číslo	Vypršení
JMS_IBM_MQMD_Feedback.	Zpětná vazba nebo kód příčiny	Celé číslo	Zpětná vazba
JMS_IBM_MQMD_Encoding.	Číselný kódování dat zprávy	Celé číslo	Kódování
JMS_IBM_MQMD_CodedCharSetId	Identifikátor znakové sady dat zprávy	Celé číslo	CodedCharSetId
JMS_IBM_MQMD_Format.	Název formátu dat zprávy	Řetězec	Formát
JMS_IBM_MQMD_Priority [1]	Priorita zprávy	Celé číslo	Priorita
JMS_IBM_MQMD_Perzistence	Trvalost zpráv	Celé číslo	Trvání
JMS_IBM_MQMD_MsgId [2]	Identifikátor zprávy	Objekt (byte []) [4]	MsgId
JMS_IBM_MQMD_CorrelId [3]	Identifikátor korelace	Objekt (byte []) [4]	CorrelId
JMS_IBM_MQMD_BackoutCount .	Počítadlo odvolaných	Celé číslo	BackoutCount
JMS_IBM_MQMD_ReplyToQ	Název fronty odpovědí	Řetězec	ReplyToQ
Správce front JMS_IBM_MQMD_ReplyTo	Název správce front odpovědí	Řetězec	ReplyToQMgr
JMS_IBM_MQMD_UserIdentifier .	Identifikátor uživatele	Řetězec	UserIdentifier
JMS_IBM_MQMD_AccountingToken .	Token evidence	Objekt (byte []) [4]	AccountingToken
JMS_IBM_MQMD_ApplIdentityData	Údaje o žádosti vztahující se k totožnosti	Řetězec	ApplIdentityData

Tabulka 126. Názvy vlastností, popisy a typy (pokračování)

Vlastnost	Popis	Javovský typ	Odkaz na úplný popis
JMS_IBM_MQMD_PutApplTyp	Typ aplikace, která vložila zprávu	Celé číslo	PutApplType
Název JMS_IBM_MQMD_PutAppl	Název aplikace, která vložila zprávu	Řetězec	PutApplName
JMS_IBM_MQMD_PutDate .	Datum, kdy byla zpráva vložena	Řetězec	PutDate
JMS_IBM_MQMD_PutTime .	Čas, kdy byla zpráva vložena	Řetězec	PutTime
JMS_IBM_MQMD_ApplOriginData	Údaje o žádosti vztahující se k původu	Řetězec	ApplOriginData
JMS_IBM_MQMD_GroupId	Identifikátor skupiny	Objekt (byte []) [4]	GroupId
JMS_IBM_MQMD_MsgSeqČíslo	Pořadové číslo logické zprávy v rámci skupiny	Celé číslo	MsgSeqNumber
JMS_IBM_MQMD_Offset.	Posunutí dat ve fyzické zprávě od začátku logické zprávy	Celé číslo	Offset
JMS_IBM_MQMD_MsgFlags	Příznaky zprávy	Celé číslo	MsgFlags
JMS_IBM_MQMD_OriginalLength	Délka původní zprávy	Celé číslo	OriginalLength

1. **Upozornění:** Přiřadíte-li hodnotu JMS_IBM_MQMD_Priority, která není v rozsahu 0-9, poruší se specifikace JMS.
2. **Upozornění:** Ve specifikaci platformy JMS je uvedeno, že ID zprávy musí být nastaveno poskytovatelem rozhraní JMS a že musí být buď jedinečné, nebo musí mít hodnotu null. Pokud přiřadíte hodnotu k JMS_IBM_MQMD_MsgId, tato hodnota se zkopíruje do JMSMessageID. Takže není nastaven poskytovatelem JMS a nemusí být jedinečná: jedná se o porušení specifikace JMS.
3. **Upozornění:** Pokud přiřadíte hodnotu JMS_IBM_MQMD_CorrelId , která začíná řetězcem 'ID:', toto porušuje specifikaci JMS.
4. **Upozornění:** Použití vlastností bajtového pole ve zprávě porušuje specifikaci platformy JMS.

Přístup k datům zpráv produktu IBM WebSphere MQ z aplikace pomocí tříd produktu WebSphere MQ pro službu JMS

K úplným datům zprávy produktu WebSphere MQ v rámci aplikace můžete přistupovat pomocí tříd produktu IBM WebSphere MQ pro platformu JMS. Chcete-li přistupovat ke všem datům, musí být tato zpráva `JMSBytesMessage`. Tělo produktu `JMSBytesMessage` obsahuje libovolné záhlaví `MQRFH2` , všechna ostatní záhlaví IBM WebSphere MQ a následující data zprávy.

Nastavte vlastnost `WMQ_MESSAGE_BODY` cíle na `WMQ_MESSAGE_BODY_MQ`, chcete-li přijmout všechna data těla zprávy v `JMSBytesMessage`.

Je-li hodnota `WMQ_MESSAGE_BODY` nastavena na hodnotu `WMQ_MESSAGE_BODY_JMS` nebo `WMQ_MESSAGE_BODY_UNSPECIFIED`, tělo zprávy se vrátí bez záhlaví JMS `MQRFH2` a vlastnosti produktu `JMSBytesMessage` odrážejí vlastnosti nastavené v produktu `RFH2`.

Některé aplikace nemohou používat funkce popsané v tomto tématu. Je-li aplikace připojena ke správci front produktu WebSphere MQ V6 nebo pokud byla nastavena hodnota `PROVIDERVERSION` do produktu 6, funkce nejsou k dispozici.

Odeslání zprávy

Při odesílání zpráv má vlastnost cíle WMQ_MESSAGE_BODYpřednost před hodnotou WMQ_TARGET_CLIENT.

Je-li hodnota WMQ_MESSAGE_BODY nastavena na hodnotu WMQ_MESSAGE_BODY_JMS, třídy WebSphere MQ pro JMS automaticky generují záhlaví MQRFH2 na základě nastavení vlastností JMSMessage a polí záhlaví.

Je-li parametr WMQ_MESSAGE_BODY nastaven na hodnotu WMQ_MESSAGE_BODY_MQ, nebude do těla zprávy přidáno žádné další záhlaví.

Je-li hodnota WMQ_MESSAGE_BODY nastavena na hodnotu WMQ_MESSAGE_BODY_UNSPECIFIED, třídy WebSphere MQ pro JMS odešlou záhlaví MQRFH2 , pokud není položka WMQ_TARGET_CLIENT nastavena na hodnotu WMQ_TARGET_DEST_MQ. Při přijetí nastavení WMQ_TARGET_CLIENT na hodnotu WMQ_TARGET_DEST_MQ způsobí, že bude z těla zprávy odebrán některý MQRFH2 .

Poznámka: JMSBytesMessage a JMSTextMessage nevyžadují MQRFH2, zatímco JMSStreamMessage, JMSMapMessage, a JMSObjectMessage .

WMQ_MESSAGE_BODY_UNSPECIFIED je výchozí nastavení pro WMQ_MESSAGE_BODYa WMQ_TARGET_DEST_JMS je výchozí nastavení pro WMQ_TARGET_CLIENT.

Pokud odešlete JMSBytesMessage, můžete přepsat výchozí nastavení těla zprávy JMS, je-li vytvořena zpráva WebSphere MQ . Použijte následující vlastnosti:

- JMS_IBM_Format nebo JMS_IBM_MQMD_Format: Tato vlastnost určuje formát záhlaví WebSphere MQ nebo informačního obsahu aplikace, který spouští tělo zprávy JMS v případě, že neexistuje žádné předchozí záhlaví produktu Websphere MQ .
- JMS_IBM_Character_Set nebo JMS_IBM_MQMD_CodedCharSetId: Tato vlastnost určuje CCSID záhlaví produktu WebSphere MQ nebo informačního obsahu aplikace, který spouští tělo zprávy JMS, pokud neexistuje předchozí záhlaví produktu Websphere MQ .
- JMS_IBM_Encoding nebo JMS_IBM_MQMD_Encoding: Tato vlastnost určuje kódování záhlaví WebSphere MQ nebo informačního obsahu aplikace, který spouští tělo zprávy JMS, pokud neexistuje předchozí záhlaví produktu Websphere MQ .

Jsou-li zadány oba typy vlastností, potlačí vlastnosti produktu JMS_IBM_MQMD_* odpovídající vlastnosti produktu JMS_IBM_* , pokud je vlastnost cíle WMQ_MQMD_WRITE_ENABLED nastavena na hodnotu true.

Rozdíly v účinku mezi nastavením vlastností zpráv pomocí JMS_IBM_MQMD_* a JMS_IBM_* jsou významné:

1. Vlastnosti JMS_IBM_MQMD_* jsou specifické pro poskytovatele JMS IBM WebSphere MQ .
2. Vlastnosti JMS_IBM_MQMD_* jsou nastaveny pouze v MQMD. Vlastnosti produktu JMS_IBM_* jsou nastaveny v produktu MQMD pouze v případě, že zpráva neobsahuje záhlaví MQRFH2 JMS. Jinak jsou nastaveny v záhlaví JMS RFH2 .
3. Vlastnosti JMS_IBM_MQMD_* nemají žádný vliv na kódování textu a čísel zapsaných do JMSMessage.

   Přijímající aplikace bude pravděpodobně předpokládat, že hodnoty MQMD.Encoding a MQMD.CodedCharSetId odpovídají kódování a znakové sadě čísel a textu v těle zprávy. Jsou-li použity vlastnosti produktu JMS_IBM_MQMD_* , je odpovědností odesílající aplikace, aby se tak stalo. Kódování a znaková sada čísel a textu v těle zprávy jsou nastaveny vlastnostmi JMS_IBM_* .

   Špatně kódovaný úsek kódu v produktu Obrázek 163 na stránce 872 odesílá zprávu kódovanou ve znakové sadě 1208 s parametrem MQMD.CodedCharSetId nastaveným na hodnotu 37.

a. Odeslat chybně kódovanou zprávu

```
TextMessage tmo = session.createTextMessage();
((MQDestination) destination).setMessageBodyStyle
                (WMQConstants.WMQ_MESSAGE_BODY_MQ);
((MQDestination)destination).setMQMDWriteEnabled(true);
tmo.setIntProperty(WMQConstants.JMS_IBM_MQMD_CODEDCHARSETID, 37);
tmo.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET, 1208);
tmo.setText("String one");
producer.send(tmo);
```

b. Příjem zprávy závisí na hodnotě parametru JMS_IBM_CHARACTER_SET nastavené hodnotou MQMD.CodedCharSetId:

```
TextMessage tmi = (TextMessage) cons.receive();
System.out.println("Message is \"" + tmi.getText() + "\"");
```

c. Výsledný výstup:

```
Message is "ëÈÊ'>...??>?"
```

Obrázek 163. Nekonzistentně kódováno MQMD a data zprávy

Jedna z úseků kódu v produktu Obrázek 164 na stránce 872 má za následek vložení zprávy do fronty nebo tématu spolu se svým tělem obsahujícím informační obsah aplikace bez automaticky generovaného záhlaví produktu MQRFH2 .

1. Nastavení WMQ_MESSAGE_BODY_MQ:

```
((MQDestination) destination).setMessageBodyStyle
                (WMQConstants.WMQ_MESSAGE_BODY_MQ);
```

2. Nastavení WMQ_TARGET_DEST_MQ:

```
((MQDestination) destination).setMessageBodyStyle
                (WMQConstants.WMQ_MESSAGE_BODY_UNSPECIFIED);
((MQDestination) destination).
                setTargetClient(WMQConstants.WMQ_TARGET_DEST_MQ);
```

Obrázek 164. Odeslat zprávu s tělem zprávy produktu MQ .

Přijímání zprávy

Je-li parametr WMQ_MESSAGE_BODY nastaven na hodnotu WMQ_MESSAGE_BODY_JMS, je příchozí typ zprávy JMS a tělo určováno obsahem přijaté zprávy produktu Websphere MQ . Typ zprávy a tělo se určují podle polí v záhlaví MQRFH2 nebo v MQMD, pokud neexistuje MQRFH2.

Je-li parametr WMQ_MESSAGE_BODY nastaven na hodnotu WMQ_MESSAGE_BODY_MQ, je příchozí typ zprávy JMS JMSBytesMessage. Tělo zprávy JMS je data zprávy vrácená základním voláním rozhraní API produktu MQGET . Délka těla zprávy je délka vrácená voláním MQGET . Znaková sada a kódování dat v těle zprávy je určeno poli CodedCharSetId a Encoding v souboru MQMD. Formát dat v těle zprávy je určen polem Formát v poli MQMD .

Je-li parametr WMQ_MESSAGE_BODY nastaven na hodnotu WMQ_MESSAGE_BODY_UNSPECIFIED, nastaví se výchozí hodnota, třídy IBM WebSphere MQ pro JMS nastaví na hodnotu WMQ_MESSAGE_BODY_JMS.

Když obdržíte JMSBytesMessage, můžete ji dekódovat pomocí odkazu na následující vlastnosti:

- JMS_IBM_Format nebo JMS_IBM_MQMD_Format: Tato vlastnost určuje formát záhlaví WebSphere MQ nebo informačního obsahu aplikace, který spouští tělo zprávy JMS v případě, že neexistuje žádné předchozí záhlaví produktu Websphere MQ .

- `JMS_IBM_Character_Set` nebo `JMS_IBM_MQMD_CodedCharSetId`: Tato vlastnost určuje `CCSID` záhlaví produktu WebSphere MQ nebo informačního obsahu aplikace, který spouští tělo zprávy JMS, pokud neexistuje předchozí záhlaví produktu Websphere MQ .
- `JMS_IBM_Encoding` nebo `JMS_IBM_MQMD_Encoding`: Tato vlastnost určuje kódování záhlaví WebSphere MQ nebo informačního obsahu aplikace, který spouští tělo zprávy JMS, pokud neexistuje předchozí záhlaví produktu Websphere MQ .

Následující úsek kódu má za následek přijatou zprávu, která je `JMSBytesMessage`. Bez ohledu na obsah přijaté zprávy a v poli formátu přijaté `MQMD`se jedná o zprávu `JMSBytesMessage`.

```
((MQDestination)destination).setMessageBodyStyle
                            (WMQConstants.WMQ_MESSAGE_BODY_MQ);
```

Cílová vlastnost WMQ_MESSAGE_BODY
WMQ_MESSAGE_BODY určuje, zda aplikace platformy JMS zpracovává MQRFH2 zprávy produktu WebSphere MQ jako součást informačního obsahu zprávy (tj. v části těla zprávy JMS).

Tabulka 127. Názvy a popisy vlastností

Vlastnost	**Krátký formát**	**Popis**
TĚLO WMQ_MESSAGE_BODY	MBODY	Určuje, zda aplikace platformy JMS zpracovává MQRFH2 zprávy produktu WebSphere MQ jako součást informačního obsahu zprávy (tj. v části těla zprávy JMS).

Tabulka 128. Názvy vlastností, hodnoty a metody nastavení			
Vlastnost	**Platné hodnoty v nástroji pro administraci (výchozí nastavení tučně)**	**Platné hodnoty v programech**	**Nastavit metodu**
WMQ_MESSAGE_BODY	• **Neuvedeno** Při odesílání produkt WebSphere MQ třídy pro službu JMS negeneruje ani negeneruje záhlaví MQRFH2 , v závislosti na hodnotě WMQ_TARGET_CLIENT. Když přijímá, působí jako hodnota JMS. • JMS Při odesílání produkt WebSphere MQ Classes for JMS automaticky vygeneruje záhlaví MQRFH2 a zahrne jej do zprávy produktu WebSphere MQ . Při příjmu třídy WebSphere MQ pro JMS nastavte vlastnosti zprávy JMS podle hodnot v záhlaví MQRFH2 (pokud existuje). Neprezentuje MQRFH2 jako část těla zprávy JMS. • MQ Při odesílání se třídy WebSphere MQ pro JMS negenerují MQRFH2. Při příjmu produkt WebSphere MQ Classes for JMS prezentuje strukturu MQRFH2 jako část těla zprávy JMS.	• **WMQ_MESSAGE_BODY_UNSPECIFIED** • WMQ_MESSAGE_BODY_BODY_JMS • WMQ_MESSAGE_BODY_MQ	setMessageBodyStyle

Trvalé zprávy JMS

Třídy WebSphere MQ pro aplikace JMS mohou používat atribut fronty produktu **NonPersistentMessageClass** k zajištění lepšího výkonu pro trvalé zprávy JMS, a to na úkor určité spolehlivosti.

Fronta produktu WebSphere MQ má atribut s názvem **NonPersistentMessageClass**. Hodnota tohoto atributu určuje, zda jsou přechodné zprávy ve frontě zahozeny, když se správce front restartuje.

Atribut pro lokální frontu můžete nastavit pomocí příkazu WebSphere MQ Script (MQSC), DEFINE QLOCAL, s některým z následujících parametrů:

TŘÍDA NPMCLASS (NORMÁLNÍ)
Netrvalé zprávy ve frontě jsou zahozeny při restartu správce front. Toto je výchozí hodnota.

TŘÍDA NPMCLASS (VYSOKÁ)
Netrvalé zprávy ve frontě se nezařadí, pokud správce front restartuje po uvedení do klidového stavu nebo okamžitého ukončení. Netrvalé zprávy mohou být vyřazeny po preventivním vypnutí nebo selhání.

Toto téma popisuje, jak třídy WebSphere MQ pro aplikace JMS mohou používat tento atribut fronty k zajištění lepšího výkonu pro trvalé zprávy JMS.

Vlastnost PERSISTENCE objektu Queue nebo Topic může mít hodnotu HIGH (vysoká). K nastavení této hodnoty můžete použít administrativní nástroj produktu WebSphere MQ JMS, nebo může aplikace volat metodu Destination.setPersistence(), která předává hodnotu WMQConstants.WMQ_PER_NPHIGH jako parametr.

Pokud aplikace odešle trvalou zprávu JMS nebo dočasnou zprávu JMS do místa určení, kde má vlastnost PERSISTENCE hodnotu HIGH, a základní fronta produktu WebSphere MQ je nastavena na hodnotu NPMCLASS (HIGH), bude zpráva vložena do fronty jako přechodná zpráva produktu WebSphere MQ . Pokud vlastnost PERSISTENCE v místě určení nemá hodnotu HIGH nebo pokud je základní fronta nastavena na hodnotu NPMCLASS (NORMAL), bude trvalá zpráva platformy JMS vložena do fronty jako trvalá zpráva produktu WebSphere MQ a fronta zpráv JMS je vložena do fronty jako přechodná zpráva produktu WebSphere MQ .

Je-li trvalá zpráva JMS vložena do fronty jako přechodná zpráva produktu WebSphere MQ a chcete se ujistit, že zpráva nebyla vyřazena po uvedení do klidového stavu nebo okamžitého ukončení práce správce front, musí být všechny fronty, přes které může být zpráva směrována, nastaveny na hodnotu NPMCLASS (HIGH). V doméně publikování/odběru tyto fronty zahrnují fronty odběratelů. Jako pomůcce pro vynucení této konfigurace WebSphere MQ Classes for JMS vydá výjimku InvalidDestination, pokud se aplikace pokusí vytvořit spotřebitele zpráv pro místo určení, kde má vlastnost PERSISTENCE hodnotu HIGH a základní fronta WebSphere MQ je nastavena na hodnotu NPMCLASS (NORMAL).

Nastavení vlastnosti PERSISTENCE na místo určení na hodnotu HIGH neovlivní způsob přijetí zprávy z daného místa určení. Zpráva odeslaná jako trvalá zpráva JMS se přijímá jako trvalá zpráva JMS a jako přechodná zpráva JMS je přijata zpráva jako přechodná zpráva JMS.

Když aplikace odešle první zprávu do místa určení, kde má vlastnost PERSISTENCE hodnotu HIGH, nebo když aplikace vytvoří první spotřebitele zpráv pro místo určení, kde má vlastnost PERSISTENCE hodnotu HIGH, třídy WebSphere MQ pro JMS vydá volání MQINQ, aby určil, zda je NPMCLASS (HIGH) nastavena na základní frontě WebSphere MQ . Žádost musí mít proto oprávnění k zjišťování informací o frontě. Kromě toho třída produktu WebSphere MQ pro službu JMS zachovává výsledek volání MQINQ, dokud nedojde k odstranění místa určení, a nevydá další volání MQINQ. Pokud tedy změníte nastavení NPMCLASS na základní frontě v době, kdy aplikace stále používá místo určení, třídy WebSphere MQ pro rozhraní JMS neoznámí nové nastavení.

Povolíte-li trvalé zprávy rozhraní JMS vkládat do front produktu WebSphere MQ jako přechodné zprávy produktu WebSphere MQ , získáte na straně nákladů výkon na úkor určité spolehlivosti. Vyžadujete-li maximální spolehlivost pro trvalé zprávy JMS, neodesílejte zprávy do místa určení, kde má vlastnost PERSISTENCE hodnotu HIGH (vysoká).

Vrstva JMS může používat SYSTEM.JMS.TEMPQ.MODEL, místo SYSTEM.DEFAULT.MODEL.QUEUE. SYSTEM.JMS.TEMPQ.MODEL vytváří trvalé dynamické fronty, které přijímají trvalé zprávy, protože SYSTEM.DEFAULT.MODEL.QUEUE nemůže přijímat trvalé zprávy. Chcete-li pro příjem trvalých zpráv použít dočasné fronty, je třeba použít příkaz SYSTEM.JMS.TEMPQ.MODEL, nebo změňte modelovou frontu na alternativní frontu dle vašeho výběru.

Použití zabezpečení SSL (Secure Sockets Layer) s třídami produktu WebSphere MQ pro službu JMS

Třídy WebSphere MQ pro aplikace JMS mohou používat šifrování SSL. K tomu potřebují poskytovatele JSSE.

Třídy WebSphere MQ pro připojení JMS používající funkci TRANSPORT (CLIENT) podporují šifrování protokolu SSL (Secure Sockets Layer). SSL poskytuje komunikační šifrování, ověření a integritu zpráv. Obvykle se používá k zabezpečení komunikace mezi dvěma rovnocennými partnery na Internetu nebo v rámci intranetu.

Třídy WebSphere MQ pro rozhraní JMS používají k ošetření šifrování SSL prostředí JSSE (Java Secure Socket Extension), a proto vyžaduje poskytovatele JSSE. Prostředí JVM JSE v1.4 má vestavěného poskytovatele JSSE. Podrobnosti o tom, jak spravovat a ukládat certifikáty se mohou lišit od

poskytovatele k poskytovateli. Další informace o tomto tématu naleznete v dokumentaci k poskytovateli JSSE.

V tomto oddílu se předpokládá, že poskytovatel JSSE je správně nainstalován a nakonfigurován a že byly nainstalovány a zpřístupněny vhodné certifikáty pro poskytovatele JSSE. Nyní můžete použít JMSAdmin k nastavení počtu administrativních vlastností.

Pokud vaše aplikace WebSphere MQ pro aplikaci JMS používá tabulku definic kanálů klienta (CCDT) k připojení ke správci front, přečtěte si téma “Použití tabulky definic kanálů klienta s třídami produktu IBM WebSphere MQ classes for JMS” na stránce 884.

Vlastnost objektu SSLCIPHERSUITE
Nastavte SSLCIPHERSUITE tak, aby bylo povoleno šifrování SSL na objektu ConnectionFactory .

Chcete-li povolit šifrování SSL na objektu ConnectionFactory , použijte JMSAdmin k nastavení vlastnosti SSLCIPHERSUITE na hodnotu CipherSuite podporovanou vaším poskytovatelem JSSE. To musí odpovídat sadě CipherSpec na cílovém kanálu. Hodnota CipherSuites se však liší od specifikace CipherSpecs , a proto mají různé názvy. Produkt “SSL CipherSpecs a CipherSuites v rozhraní JMS” na stránce 879 obsahuje tabulku obsahující mapování CipherSpecs podporované produktem WebSphere MQ na ekvivalentní sadu CipherSuites , která je známa jako rozšíření JSSE. Další informace o sadách CipherSpecs a CipherSuites k produktu WebSphere MQnaleznete v příručce Zabezpečení.

Chcete-li například nastavit objekt ConnectionFactory , který lze použít k vytvoření připojení přes kanál MQI s povoleným SSL se sadou CipherSpec RC4_MD5_EXPORT, zadejte pro správce JMSAdmin následující příkaz:

```
ALTER CF(my.cf) SSLCIPHERSUITE(SSL_RSA_EXPORT_WITH_RC4_40_MD5)
```

To lze také nastavit z aplikace pomocí metody setSSLCipherSuite () na objektu MQConnectionFactory .

Pokud je pro usnadnění práce zadána vlastnost CipherSpec v rámci vlastnosti SSLCIPHERSUITE, pokusí se služba JMSAdmin mapovat položku CipherSpec na příslušnou sadu CipherSuite a vydá varovnou zprávu. Tento pokus o mapování se nemapuje, je-li vlastnost určena aplikací.

Po prvním použití použijte tabulku CCDT (Client Channel Definition Table). Další informace viz “Použití tabulky definic kanálů klienta s třídami produktu IBM WebSphere MQ classes for JMS” na stránce 884.

Vlastnost objektu SSLFIPSREQUIRED
Pokud vyžadujete připojení pro použití sady CipherSuite , které je podporováno poskytovatelem IBM Java JSSE FIPS (IBMJSSEFIPS), nastavte vlastnost SSLFIPSREQUIRED na továrnu připojení na hodnotu YES.

Výchozí hodnota této vlastnosti je NO, což znamená, že připojení může používat libovolnou sadu CipherSuite , kterou podporuje produkt WebSphere MQ.

Pokud aplikace používá více než jedno připojení, hodnota SSLFIPSREQUIRED, která se použije, když aplikace vytvoří první připojení, určuje hodnotu, která se použije, když aplikace vytvoří jakékoli následné připojení. To znamená, že hodnota vlastnosti SSLFIPSREQUIRED továrny připojení použitá k vytvoření následného připojení je ignorována. Musíte restartovat aplikaci, chcete-li použít jinou hodnotu SSLFIPSREQUIRED.

Aplikace může tuto vlastnost nastavit voláním metody setSSLFipsRequired () objektu ConnectionFactory . Vlastnost je ignorována, pokud není nastavena žádná sada CipherSuite .

Související úlohy
Určení, že pro běhové prostředí klienta MQI je použit pouze certifikovaný standard FIPS CipherSpecs
Související odkazy
Federální standardy zpracování informací (FIPS) pro UNIX, Linux a Windows

Vlastnost objektu SSLPEERNAME
Pomocí parametru SSLPEERNAME zadejte vzor rozlišujícího názvu, abyste se ujistili, že se aplikace rozhraní JMS připojuje ke správnému správci front.

Aplikace JMS může zajistit, aby se připojovala ke správnému správci front zadáním vzoru rozlišujícího názvu (DN). Připojení je úspěšné pouze v případě, že správce front představuje DN, které odpovídá vzoru. Další podrobnosti o formátu tohoto vzoru naleznete v souvisejících tématech.

Rozlišující název je nastaven pomocí vlastnosti SSLPEERNAME objektu ConnectionFactory . Například následující příkaz JMSAdmin nastaví objekt ConnectionFactory tak, aby očekával, že se správce front bude identifikovat s názvem Common Name začínajícím znaky `QMGR.` a s alespoň dvěma názvy organizačních jednotek, přičemž první z nich musí být IBM a druhý WEBSPHERE:

```
ALTER CF(my.cf) SSLPEERNAME(CN=QMGR.*, OU=IBM, OU=WEBSPHERE)
```

Při zaškrtnutí se nerozlišují velká a malá písmena a středníky lze použít místo čárek. SSLPEERNAME lze také nastavit z aplikace pomocí metody setSSLPeerName () na objektu MQConnectionFactory . Není-li tato vlastnost nastavena, nebude u rozlišujícího názvu zadaného správcem front provedena žádná kontrola. Tato vlastnost je ignorována, pokud není nastavena žádná sada CipherSuite .

Vlastnost objektu SSLCERTSTORES
Použijte SSLCERTSTORES k uvedení seznamu serverů LDAP, které se mají použít pro kontrolu seznamu odvolaných certifikátů (CRL).

Je běžné použít seznam odvolaných certifikátů (CRL) k identifikaci certifikátů, které již nejsou důvěryhodné. Seznamy CRL jsou obvykle hostovány na serverech LDAP. Platforma JMS umožňuje, aby byl server LDAP určen pro kontrolu CRL v prostředí Java 2 v1.4 nebo novější. Následující příklad JMSAdmin nasměruje platformu JMS k použití seznamu CRL hostovaného na serveru LDAP s názvem crl1.ibm.com:

```
ALTER CF(my.cf) SSLCRL(ldap://crl1.ibm.com)
```

Poznámka: Chcete-li produkt CertStore úspěšně používat s názvem CRL hostovaným na serveru LDAP, ujistěte se, že sada SDK (Java Software Development Kit) je kompatibilní se seznamem CRL. Některé sady SDK vyžadují, aby seznam CRL odpovídal RFC 2587, které definuje schéma pro protokol LDAP v2. Většina serverů LDAP v3 používá místo toho RFC 2256.

Pokud váš server LDAP není spuštěn na výchozím portu 389, můžete jej zadat připojením dvojtečky (:) a čísla portu k názvu hostitele. Pokud se certifikát prezentovaný správcem front nachází v seznamu odvolaných certifikátů hostovaném na serveru crl1.ibm.com, připojení není dokončeno. Chcete-li se vyhnout jednomu bodu selhání, služba JMS umožňuje dodat více serverů LDAP zadáním seznamu serverů LDAP oddělených znakem mezery. Příklad:

```
ALTER CF(my.cf) SSLCRL(ldap://crl1.ibm.com ldap://crl2.ibm.com)
```

Je-li určeno více serverů LDAP, služba JMS se postupně pokusí o jednu z těchto serverů, dokud nenajde server, na kterém může úspěšně ověřit certifikát správce front. Každý server musí obsahovat stejné informace.

Řetězec v tomto formátu může být dodán aplikací na metodě MQConnectionFactory.setSSLCertStores (). Alternativně může aplikace vytvořit jeden nebo více objektů java.security.cert.CertStore , umístit je do vhodného objektu kolekce a tento objekt kolekce dodat metodě setSSLCertStores (). Tímto způsobem může aplikace upravit kontrolu CRL. Podrobné informace o vytváření a používání objektů CertStore naleznete v dokumentaci k prostředí JSSE.

Certifikát, který předkládá správce front při nastavení připojení, je ověřen následujícím způsobem:

1. První objekt CertStore v kolekci identifikovaný pomocí úložišť sslCertse používá k identifikaci serveru CRL.
2. Došlo k pokusu o kontaktování serveru CRL.
3. Je-li pokus úspěšný, prohledá se server na shodu certifikátu.
    a. Pokud má být certifikát odvolán, proces vyhledávání skončil a žádost o připojení selže s kódem příčiny MQRC_SSL_CERTIFICATE_REVOKED.

b. Pokud certifikát nebyl nalezen, je proces vyhledávání znovu spuštěn a připojení je povoleno pokračovat.

4. Je-li pokus o kontaktování serveru neúspěšný, bude použit další objekt CertStore pro identifikaci serveru CRL a proces se opakuje z kroku 2.

   Pokud se jednalo o poslední CertStore v kolekci, nebo pokud kolekce neobsahuje žádné objekty CertStore , došlo k selhání procesu vyhledávání a žádost o připojení se nezdařila s kódem příčiny MQRC_SSL_CERT_STORE_ERROR.

Objekt kolekce určuje pořadí, ve kterém jsou použita položka CertStores .

Pokud vaše aplikace používá úložiště setSSLCertStores () k nastavení kolekce objektů CertStore , nebude již objekt MQConnectionFactory svázán s oborem názvů JNDI. Pokus o to, aby to udělal, způsobuje výjimku. Pokud není nastavena vlastnost sslCertStores, neprovede se kontrola odvolání v certifikátu poskytnutém správcem front. Tato vlastnost je ignorována, pokud není nastavena žádná sada CipherSuite .

Vlastnost objektu SSLRESETCOUNT
Tato vlastnost představuje celkový počet bajtů odeslaných a přijatých připojením před opětovným získáním tajného klíče, který je použit pro šifrování.

Počet odeslaných bajtů je číslo před šifrováním a počet přijatých bajtů je číslo po dešifrování. Počet bajtů zahrnuje také řídicí informace odeslané a přijaté třídami produktu WebSphere MQ pro JMS.

Chcete-li například konfigurovat objekt ConnectionFactory , který lze použít k vytvoření připojení prostřednictvím kanálu MQI s povoleným SSL s použitím tajného klíče, který je znovu vyjednán po 4 MB dat, zadejte do správce JMSAdmin tento příkaz:

```
ALTER CF(my.cf) SSLRESETCOUNT(4194304)
```

Aplikace může tuto vlastnost nastavit voláním metody Count () setSSLReset() objektu ConnectionFactory .

Je-li hodnota této vlastnosti nula, což je výchozí hodnota, nebude tajný klíč nikdy znovu vyjednáván. Vlastnost je ignorována, pokud není nastavena žádná sada CipherSuite .

Vlastnost objektu SSLSocketFactory
Chcete-li upravit další aspekty připojení SSL pro aplikaci, vytvořte objekt SSLSocketFactory a nakonfigurujte platformu JMS tak, aby ji používala.

Možná budete chtít upravit i jiné aspekty připojení SSL pro aplikaci. Například můžete chtít inicializovat kryptografický hardware nebo změnit úložiště klíčů a úložiště údajů o důvěryhodnosti, které používáte. Chcete-li to provést, aplikace musí nejprve vytvořit objekt javax.net.ssl.SSLSocketFactory , který je odpovídajícím způsobem upraven. Informace o tom, jak to provést, jsou uvedeny v dokumentaci k prostředí JSSE, protože přizpůsobitelné funkce se liší od poskytovatele k poskytovateli. Po získání vhodného objektu SSLSocketFactory použijte metodu MQConnectionFactory.setSSLSocketFactory () pro konfiguraci platformy JMS pro použití přizpůsobeného objektu SSLSocketFactory .

Pokud vaše aplikace používá metodu setSSLSocketFactory () k nastavení přizpůsobeného objektu SSLSocketFactory , objekt MQConnectionFactory již nelze svázat s oborem názvů JNDI. Pokus o to, aby to udělal, způsobuje výjimku. Není-li tato vlastnost nastavena, použije se výchozí objekt SSLSocketFactory . Podrobnosti o chování výchozího objektu SSLSocketFactory najdete v dokumentaci k prostředí JSSE. Tato vlastnost je ignorována, pokud není nastavena žádná sada CipherSuite .

Důležité: Nepředpokládejte, že použití vlastností SSL zajišťuje zabezpečení, je-li objekt ConnectionFactory načten z oboru názvů JNDI, který není sám o sobě zabezpečený. Specificky není standardní implementace LDAP rozhraní JNDI zabezpečena. Útočník může imitovat server LDAP a uvést aplikaci JMS tak, aby se připojovala k chybnému serveru, aniž by si všimla. Jsou-li zajištěny vhodné mechanismy zabezpečení, jsou zabezpečeny další implementace rozhraní JNDI (jako je implementace fscontext).

Provedení změn v úložišti klíčů nebo úložišti údajů o důvěryhodnosti JSSE
Provedete-li změny v úložišti klíčů nebo úložišti údajů o důvěryhodnosti, musíte provést určité akce, aby změny byly vyzvednuty.

Pokud změníte obsah úložiště klíčů nebo úložiště údajů o důvěryhodnosti prostředí JSSE nebo změníte umístění souboru úložiště klíčů nebo úložiště údajů o důvěryhodnosti, třídy produktu WebSphere MQ pro aplikace JMS, které jsou spuštěny v daném okamžiku, nebudou automaticky vyzvedne provedené změny. Aby se změny projevily, musí být provedeny následující akce:

- Aplikace musí zavřít všechna svá připojení a zničit veškerá nepoužívaná připojení ve fondech připojení.
- Pokud poskytovatel JSSE ukládá informace z úložiště klíčů a úložiště údajů o důvěryhodnosti do mezipaměti, musí být tyto informace aktualizovány.

Po provedení těchto akcí mohou aplikace znovu vytvořit svá připojení.

V závislosti na návrhu aplikací a na funkci poskytované vaším poskytovatelem JSSE může být možné provést tyto akce bez zastavení a restartování aplikací. Avšak zastavení a restartování aplikací může být nejjednodušším řešením.

SSL CipherSpecs a CipherSuites v rozhraní JMS
CipherSpecs podporované produktem WebSphere MQ a jejich ekvivalenty CipherSuites.

Produkt Tabulka 129 na stránce 879 vypíše seznam CipherSpecs podporovaných produktem WebSphere MQ a jejich ekvivalenty CipherSuites. Je-li vlastnost SSLFIPSREQUIRED pro vlastnost ConnectionFactory nastavena na hodnotu NO, může se aplikace WebSphere MQ pro aplikaci JMS připojit ke správci front, pokud je na konci serveru MQI zadána jakákoli podporovaná sada CipherSpec a ekvivalentní sada CipherSuite na straně klienta je zadána. Je-li parametr SSLFIPSREQUIRED nastaven na hodnotu YES, určuje kombinace hodnoty CipherSpec a CipherSuite , zda se aplikace může připojit ke správci front.

Na konci kanálu MQI lze název CipherSpec zadat jako hodnotu parametru SSLCIPH u příkazu DEFINE CHANNEL CHLTYPE (SVRCONN). Na konci klienta kanálu MQI lze název sady CipherSuite zadat následujícími způsoby:

- Aplikace může volat metodu setSSLCipherSuite () objektu ConnectionFactory .
- Pomocí nástroje pro administraci produktu WebSphere MQ JMS můžete nastavit vlastnost SSLCIPHERSUITE objektu ConnectionFactory .

Tabulka 129. CipherSpecs podporované produktem WebSphere MQ a jejich ekvivalenty CipherSuites

CipherSpec	**Ekvivalentní CipherSuite**	**Je možné připojení, pokud je SFIPS [1] nastavena na YES?**
NULL_MD5	SSL_RSA_WITH_NULL_MD5	Ne
NULL_SHA	SSL_RSA_WITH_NULL_SHA	Ne
RC4_MD5_EXPORT	SSL_RSA_EXPORT_WITH_RC4_40_MD5	Ne
RC4_MD5_US	SSL_RSA_WITH_RC4_128_MD5	Ne
RC4_SHA_US	SSL_RSA_WITH_RC4_128_SHA	Ne
RC2_MD5_EXPORT	SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5	Ne
DES_SHA_EXPORT	SSL_RSA_WITH_DES_CBC_SHA	Ne
RC4_56_SHA_EXPORT1024	SSL_RSA_EXPORT1024_WITH_RC4_56_SHA	Ne
DES_SHA_EXPORT1024	SSL_RSA_EXPORT1024_WITH_DES_CBC_SHA	Ne
TRIPLE_DES_SHA_US	SSL_RSA_WITH_3DES_EDE_CBC_SHA	Ne

Tabulka 129. CipherSpecs podporované produktem WebSphere MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	Ekvivalentní CipherSuite	Je možné připojení, pokud je SFIPS [1] nastavena na YES?
TLS_RSA_WITH_NULL_SHA256	SSL_RSA_WITH_NULL_SHA256	Ne [7]
TLS_RSA_WITH_AES_128_CBC_SHA	SSL_RSA_WITH_AES_128_CBC_SHA	Ano [5] [7]
TLS_RSA_WITH_AES_256_CBC_SHA	SSL_RSA_WITH_AES_256_CBC_SHA	Ano [5] [7]
TLS_RSA_WITH_AES_256_CBC_SHA256	SSL_RSA_WITH_AES_256_CBC_SHA256	Ano [5] [7]
AES_SHA_US [2]		
TLS_RSA_WITH_DES_CBC_SHA [8] [9]	SSL_RSA_WITH_DES_CBC_SHA	Ne [3]
TLS_RSA_WITH_3DES_EDE_CBC_SHA [8]	SSL_RSA_WITH_3DES_EDE_CBC_SHA	Ano
FIPS_WITH_DES_CBC_SHA	SSL_RSA_FIPS_WITH_DES_CBC_SHA	Ne [4]
FIPS_WITH_3DES_EDE_CBC_SHA	SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA	Ne [6]

Notes:

1. Při použití nástroje pro administraci produktu WebSphere MQ JMS je SFIPS krátký název vlastnosti SSLFIPSREQUIRED pro vlastnost ConnectionFactory .
2. Tato CipherSpec nemá ekvivalent CipherSuite.
3. Tato CipherSpec byla certifikována FIPS 140-2 certifikovaná před 19th . květnem 2007.
4. Tato CipherSpec byla certifikována FIPS 140-2 certifikovaná před 19th . květnem 2007. Název FIPS_WITH_DES_CBC_SHA je historický a odráží fakt, že tato CipherSpec byla již dříve (ale již není) kompatibilní s FIPS-. Tato specifikace šifrování byla zamítnuta a její použití se nedoporučuje.
5. Tyto CipherSpecs (TLS_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA256) nelze použít k zabezpečení připojení z produktu WebSphere MQ Explorer ke správci front, pokud nejsou použity příslušné soubory neomezených zásad na prostředí JRE používané průzkumníkem.

   Další informace o souborech zásad naleznete v tématu Informace o zabezpečení .
6. Název FIPS_WITH_3DES_EDE_CBC_SHA je historický a odráží fakt, že tato CipherSpec již byla (ale již není delší) kompatibilní s FIPS-. Tato specifikace šifrování byla zamítnuta a její použití se nedoporučuje.
7. Tyto CipherSpecs (TLS_RSA_WITH_NULL_SHA256, TLS_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA256) vyžadují prostředí IBM JRE 6.0 SR13 FP2 , 7.0 SR4 FP2 nebo novější.
8. Tyto CipherSpecs (TLS_RSA_WITH_3DES_EDE_CBC_SHA, TLS_RSA_WITH_DES_CBC_SHA, TLS_RSA_WITH_RC4_128_SHA256) mohou používat buď SSLv3 , nebo TLS. Je-li standard FIPS standardně povolen, použije se SSLv3 standardně. Chcete-li použít TLS, nastavte systémovou vlastnost Java **com.ibm.mq.cfg.preferTLS** na `true`.
9. Tato CipherSpec TLS_RSA_WITH_3DES_EDE_CBC_SHA byla zamítnuta. Nicméně lze ji přesto použít k přenosu až 32 GB dat před ukončením připojení s chybou AMQ9288. Chcete-li se této chybě vyhnout, je třeba při použití této CipherSpecbuď zabránit použití trojitého DES, nebo povolit resetování tajného klíče.

Související informace

Určení, že pro běhové prostředí klienta MQI je použit pouze certifikovaný standard FIPS CipherSpecs

Probíhá zápis kanálů kanálů v jazyce Java pro třídy WebSphere MQ pro JMS

Uživatelské procedury kanálu vytvoříte definováním tříd Java, které implementují zadaná rozhraní.

Tři rozhraní jsou definována v balíku com.ibm.mq.exits :

- WMQSendExitpro ukončení odeslání
- WMQReceiveExit, pro uživatelskou proceduru pro příjem
- WMQSecurityExitpro uživatelskou proceduru pro zabezpečení zprávy

Následující ukázka kódu definuje třídu, která implementuje všechna tři rozhraní:

```
public class MyMQExits implements
WMQSendExit, WMQReceiveExit, WMQSecurityExit {
    // Default constructor
  public MyMQExits(){
  }
    // This method implements the send exit interface
  public ByteBuffer channelSendExit(
MQCXP channelExitParms,
                                  MQCD channelDefinition,
                                  ByteBuffer agentBuffer)
  {
    // Complete the body of the send exit here
  }
    // This method implements the receive exit interface
  public ByteBuffer channelReceiveExit(
MQCXP channelExitParms,
                                     MQCD channelDefinition,
                                     ByteBuffer agentBuffer)
  {
    // Complete the body of the receive exit here
  }
    // This method implements the security exit interface
  public ByteBuffer channelSecurityExit(
MQCXP channelExitParms,
                                      MQCD channelDefinition,
                                      ByteBuffer agentBuffer)
  {
    // Complete the body of the security exit here
  }
}
```

Každá uživatelská procedura obdrží jako parametry objekt MQCXP a objekt MQCD. Tyto objekty reprezentují struktury MQCXP a MQCD definované v procedurálním rozhraní.

Je-li zavolána uživatelská procedura pro odeslání zprávy, obsahuje parametr agentBuffer data, která se mají odeslat do správce front serveru. Parametr délky není povinný, protože výraz agentBuffer.limit () poskytuje délku dat. Uživatelská procedura odeslání vrátí jako hodnotu data, která mají být odeslána správci front serveru. Pokud však uživatelská procedura odeslání není poslední uživatelskou procedurou odeslání v posloupnosti uživatelských procedur odeslání, vrátí se vrácená data místo na další uživatelskou proceduru odeslání v posloupnosti. Uživatelská procedura odeslání může vrátit upravenou verzi dat, která přijme, v parametru agentBuffer , nebo může vrátit data nezměněná. Nejjednodušším možným výstupním tělem je proto:

```
{ return agentBuffer; }
```

Je-li volána uživatelská procedura pro přijetí zprávy, obsahuje parametr agentBuffer data, která byla přijata ze správce front serveru. Uživatelská procedura pro příjem vrátí data, která mají být předána třídám produktu WebSphere MQ pro platformu JMS, jako hodnotu. Pokud však uživatelská procedura pro přijetí není poslední uživatelskou procedurou příjmu v posloupnosti uživatelských procedur příjmu, vrátí se vrácená data místo na další uživatelskou proceduru pro přijetí v posloupnosti.

Když je zavolána uživatelská procedura zabezpečení, obsahuje parametr agentBuffer data, která byla přijata v toku zabezpečení z uživatelské procedury zabezpečení na konci připojení serveru. Uživatelská

procedura zabezpečení se vrátí jako hodnota dat, která mají být odeslána v rámci zabezpečení serveru, do uživatelské procedury zabezpečení serveru.

Uživatelské procedury kanálu jsou volány s vyrovnávací pamětí, která má záložní pole. Pro nejlepší výkon by měla uživatelská procedura vrátit vyrovnávací paměť s záložním polem.

Až bude voláno do uživatelské procedury kanálu, může být předáno až 32 znaků uživatelských dat. Uživatelská procedura přistupuje k datům uživatele voláním metody getExitData () objektu MQCXP. Ačkoli uživatelská procedura může změnit uživatelská data voláním metody setExitData (), uživatelská data se obnoví při každém vyvolání procedury ukončení. Jakékoli změny provedené v uživatelských datech se proto ztratí. Uživatelská procedura však může předávat data z jednoho volání do dalšího pomocí oblasti uživatelských procedur objektu MQCXP. Uživatelská procedura přistupuje k uživatelské oblasti uživatelské procedury, a to voláním metody getExitUserArea().

Každá výstupní třída musí mít konstruktor. Konstruktorem může být buď výchozí konstruktor, jak je zobrazeno v předchozím příkladu, nebo konstruktor s řetězcovým parametrem. Konstruktor je volán k vytvoření instance třídy ukončení pro každou uživatelskou proceduru definovanou ve třídě. Proto je v předchozím příkladu pro uživatelskou proceduru pro odeslání vytvořena instance třídy MyMQExits , pro uživatelskou proceduru pro příjem je vytvořena jiná instance a pro uživatelskou proceduru zabezpečení je vytvořena třetí instance. Když je volán konstruktor s řetězcovým parametrem, parametr obsahuje stejná uživatelská data, která jsou předána uživatelské proceduře kanálu, pro kterou je instance vytvářena. Pokud má třída ukončení jak výchozí konstruktor, tak konstruktor s jedním parametrem, má přednost konstruktor jednoho parametru.

Nezavírejte připojení z uživatelské procedury kanálu.

Když se data odesílají na konec připojení serveru, je šifrování SSL provedeno *za* , jsou volány všechny uživatelské procedury kanálu. Podobně platí, že když jsou data přijata ze konce připojení serveru, dešifrování SSL se provádí *před* voláním jakýchkoli uživatelských procedur kanálu.

Ve verzích produktu WebSphere MQ pro systém JMS starších než verze 7.0byly uživatelské procedury kanálu implementovány pomocí rozhraní MQSendExit, MQReceiveExita MQSecurityExit. Tato rozhraní můžete i nadále používat, ale nová rozhraní jsou upřednostňována pro zlepšení funkcí a výkonu.

Konfigurace produktu IBM WebSphere MQ classes for JMS pro použití uživatelských procedur kanálu

Aplikace produktu IBM WebSphere MQ classes for JMS může v kanálu MQI, který se spustí při připojení aplikace ke správci front, použít zabezpečení kanálu, odeslání a přijetí. Aplikace může používat uživatelské procedury zapsané v Java, C nebo C + +. Aplikace může také použít posloupnost uživatelských procedur pro odesílání nebo příjem, které jsou spouštěny za dědění.

Následující vlastnosti se používají k určení uživatelské procedury odeslání nebo posloupnosti uživatelských procedur pro odesílání zpráv používaných připojením produktu JMS :

- Vlastnost **SENDEXIT** objektu MQConnectionFactory .
- Vlastnost **sendexit** ve specifikaci aktivace, kterou používá adaptér prostředků IBM WebSphere MQ pro příchozí komunikaci,
- Vlastnost **sendexit** na objektu ConnectionFactory , kterou používá adaptér prostředků IBM WebSphere MQ pro výstupní komunikaci.

Hodnota vlastnosti je řetězec, který obsahuje jednu nebo více položek oddělených čárkami. Každá položka identifikuje uživatelskou proceduru pro odeslání zprávy jedním z následujících způsobů:

- Název třídy, která implementuje rozhraní `WMQSendExit` pro uživatelskou proceduru pro odeslání zprávy, která je zapsána v souboru Java.
- Řetězec ve formátu *libraryName* (*entryPointName*) pro uživatelskou proceduru pro odeslání zprávy v jazyce C nebo C + +.

Podobným způsobem určují následující vlastnosti uživatelskou proceduru pro přijetí zprávy nebo posloupnost uživatelských procedur pro příjem, kterou používá připojení:

- Vlastnost **RECEXIT** objektu MQConnectionFactory .

- Vlastnost **receiveexit** ve specifikaci aktivace, kterou používá adaptér prostředků IBM WebSphere MQ pro příchozí komunikaci,
- Vlastnost **receiveexit** na objektu ConnectionFactory , kterou používá adaptér prostředků IBM WebSphere MQ pro výstupní komunikaci.

Následující vlastnosti určují proceduru zabezpečení použitou při připojení:

- Vlastnost **SECEXIT** objektu MQConnectionFactory .
- Vlastnost **securityexit** ve specifikaci aktivace, kterou používá adaptér prostředků IBM WebSphere MQ pro příchozí komunikaci,
- Vlastnost **securityexit** na objektu ConnectionFactory , kterou používá adaptér prostředků IBM WebSphere MQ pro výstupní komunikaci.

V případě MQConnectionFactoriesmůžete nastavit vlastnosti **SENDEXIT**, **RECEXIT** a **SECEXIT** pomocí nástroje pro administraci produktu IBM WebSphere MQ JMS nebo IBM WebSphere MQ Explorer. Alternativně může aplikace nastavit vlastnosti voláním metod `setSendExit()`, `setReceiveExit()`a `setSecurityExit()` .

Uživatelské procedury kanálu jsou načítány vlastním zavaděčem tříd. Chcete-li vyhledat uživatelskou proceduru kanálu, prohledávač tříd prohledá následující umístění v uvedeném pořadí.

1. Cesta ke třídě určená vlastností **com.ibm.mq.cfg.ClientExitPath.JavaExitsClasspath** nebo atributem **JavaExitsClassPath** v sekci Kanály konfiguračního souboru klienta IBM WebSphere MQ .
2. Cesta ke třídě určená systémovou vlastností Java **com.ibm.mq.exitClasspath**. Mějte na zřeteli, že tato vlastnost je nyní zamítnuta.
3. Adresář IBM WebSphere MQ opustí adresář, jak ukazuje Tabulka 130 na stránce 883. Zavaděč tříd nejprve prohledá adresář pro soubory tříd, které nejsou zabaleny v souborech archivu produktu Java (JAR). Není-li nalezena uživatelská procedura kanálu, bude zavaděč tříd poté hledat v souborech JAR v adresáři.

Tabulka 130. Adresář uživatelských procedur IBM WebSphere MQ

Platforma	**Adresář**
UNIX and Linux	`/var/mqm/exits` (32bitové uživatelské procedury kanálu) `/var/mqm/exits64` (uživatelské procedury 64bitového kanálu)
Windows	*instalační_dat_adr*\exits kde *instalační_dat_adr* je adresář, který jste vybrali pro datové soubory produktu IBM WebSphere MQ během instalace. Standardní adresář je `C:\Program Files\IBM\WebSphere MQ`.

Poznámka: Pokud uživatelská procedura kanálu existuje ve více než jednom umístění, načte produkt IBM WebSphere MQ classes for JMS první nalezenou instanci.

Nadřízeným objektem zavaděče tříd je zaváděč tříd, který se používá k načtení produktu IBM WebSphere MQ classes for JMS. Je tedy možné, aby nadřízený zavaděč tříd zavedl uživatelskou proceduru kanálu, pokud ji nelze najít v žádném z předchozích umístění. Pokud však používáte produkt IBM WebSphere MQ classes for JMS v prostředí, jako je například aplikační server JEE , nebudete pravděpodobně moci ovlivnit volbu nadřazeného zavaděče tříd, a proto by měl být zavaděč tříd konfigurován nastavením systémové vlastnosti Java **com.ibm.mq.cfg.ClientExitPath.JavaExitsClasspath** na aplikačním serveru.

Pokud je vaše aplikace spuštěna se zapnutým produktem Java Security Manager, pak musí mít konfigurační soubor zásad používaný běhovým prostředím Java , v němž je spuštěna aplikace, oprávnění k zavedení třídy uživatelské procedury kanálu. Informace o tom, jak to provést, najdete v tématu Spuštění tříd produktu IBM MQ pro aplikace JMS pod produktem Java Security Manager.

Rozhraní `MQSendExit`, `MQReceiveExita` `MQSecurityExit` dodaná s verzemi IBM WebSphere MQ staršími než Version 7.0 jsou stále podporovány. Pokud používáte uživatelské procedury kanálu, které implementují tato rozhraní, musí být v cestě ke třídě uvedena hodnota `com.ibm.mq.jar` .

Informace o tom, jak zapisovat uživatelské procedury kanálu v jazyce C, najdete v tématu "Kanály-uživatelské programy pro kanály systému zpráv" na stránce 380. Musíte uložit uživatelské programy kanálu napsané v jazycích C nebo C++ v adresáři zobrazeném v produktu Tabulka 130 na stránce 883.

Pokud vaše aplikace používá tabulku definic kanálů klienta (CCDT) k připojení ke správci front, prohlédněte si téma "Použití tabulky definic kanálů klienta s třídami produktu IBM WebSphere MQ classes for JMS" na stránce 884.

Určení uživatelských dat, která mají být předána uživatelským procedurám kanálu při použití tříd WebSphere MQ pro JMS

Až bude voláno do uživatelské procedury kanálu, může být předáno až 32 znaků uživatelských dat.

Vlastnost SENDEXITINIT objektu MQConnectionFactory určuje uživatelská data, která jsou předávána každému ukončovacímu programu odeslání při volání. Hodnota vlastnosti je řetězec, který obsahuje jednu nebo více položek dat uživatele oddělených čárkami. Pozice každé položky uživatelských dat v rámci řetězce určuje, který výstup odeslání bude v posloupnosti uživatelských procedur pro odesílání předán. Například první položka uživatelských dat v řetězci se předá do první uživatelské procedury odeslání v posloupnosti uživatelských procedur odeslání.

Vlastnost SENDEXITINIT můžete nastavit pomocí nástroje pro administraci produktu WebSphere MQ JMS nebo Průzkumníka WebSphere MQ . Alternativně může aplikace nastavit tuto vlastnost voláním metody setSendExitInit().

Podobně vlastnost RECEXITINIT objektu ConnectionFactory určuje uživatelská data předávaná pro každou uživatelskou proceduru pro přijetí zprávy a vlastnost SECEXITINIT určuje uživatelská data předávaná uživatelské proceduře zabezpečení. Tyto vlastnosti můžete nastavit pomocí nástroje pro administraci produktu WebSphere MQ JMS nebo Průzkumníka WebSphere MQ . Alternativně může aplikace nastavit vlastnosti voláním metod setReceiveExitInit() a setSecurityExitInit().

Všimněte si následujících pravidel, když uvádíte uživatelská data, která jsou předána uživatelským procedurám kanálu:

- Pokud je počet položek uživatelských dat v řetězci více než počet uživatelských procedur v posloupnosti, přebytečné položky uživatelských dat se budou ignorovat.
- Pokud je počet položek uživatelských dat v řetězci menší než počet uživatelských procedur v posloupnosti, každá nespecifikovaná položka uživatelských dat je nastavena na prázdný řetězec. Dva čárky za sebou v řetězci, nebo čárka na začátku řetězce, také označují neuvedenou položku uživatelských dat.

Pokud aplikace používá tabulku CCDT (Client Channel Definition CCDT) k připojení ke správci front, budou veškerá uživatelská data zadaná v definici kanálu připojení klienta předána uživatelským procedurám kanálu při jejich volání. Další informace o použití tabulky definic kanálů klienta viz "Použití tabulky definic kanálů klienta s třídami produktu IBM WebSphere MQ classes for JMS" na stránce 884.

Použití tabulky definic kanálů klienta s třídami produktu IBM WebSphere MQ classes for JMS

Třídy IBM WebSphere MQ pro aplikaci JMS mohou používat definice kanálů připojení klienta, které jsou uloženy v tabulce CCDT (Client Channel Definition table). Objekt ConnectionFactory nakonfigurujete pro použití tabulky CCDT. Existují některá omezení jejího používání.

Jako alternativu k vytvoření definice kanálu připojení klienta pomocí nastavení určitých vlastností objektu ConnectionFactory může aplikace IBM WebSphere MQ classes for JMS používat definice kanálů připojení klienta, které jsou uloženy v tabulce definic kanálů klienta. Tyto definice jsou vytvořeny pomocí příkazů skriptu IBM WebSphere MQ Script (MQSC) nebo IBM WebSphere MQ Programmable Command Format (PCF). Když aplikace vytvoří objekt připojení, produkt IBM WebSphere MQ classes for JMS prohledá tabulku definic kanálů klienta pro vhodnou definici kanálu připojení klienta a použije definici kanálu ke

spuštění kanálu MQI. Další informace o tabulkách definic kanálů klienta a o tom, jak je vytvořit, najdete v tématu Tabulka definic kanálů klienta.

Chcete-li použít tabulku definic kanálů klienta, vlastnost CCDTURL objektu ConnectionFactory musí být nastavena na objekt adresy URL. Objekt URL zapouzdřuje adresu URL, která identifikuje název a umístění souboru obsahujícího tabulku definic kanálů klienta a určuje, jak lze k souboru přistupovat. Vlastnost CCDTURL můžete nastavit pomocí nástroje pro administraci produktu IBM WebSphere MQ JMS nebo můžete tuto vlastnost nastavit tak, že vytvoříte objekt adresy URL a zavolá metodu setCCDTURL() objektu ConnectionFactory .

Pokud například soubor ccdt1.tab obsahuje tabulku definic kanálů klienta a je uložen ve stejném systému, v němž je aplikace spuštěna, může aplikace nastavit vlastnost CCDTURL následujícím způsobem:

```
java.net.URL chanTab1 = new URL("file:///home/admdata/ccdt1.tab");
factory.setCCDTURL(chanTab1);
```

Jako další příklad předpokládejme, že soubor ccdt2.tab obsahuje tabulku definic kanálů klienta a je uložen v systému, který se liší od tabulky, na které je aplikace spuštěna. Je-li k souboru přístup pomocí protokolu FTP, může aplikace nastavit vlastnost CCDTURL následujícím způsobem:

```
java.net.URL chanTab2 = new URL("ftp://ftp.server/admdata/ccdt2.tab");
factory.setCCDTURL(chanTab2);
```

Kromě nastavení vlastnosti CCDTURL objektu ConnectionFactory , musí být vlastnost QMANAGER stejného objektu nastavena na jednu z následujících hodnot:

- Název správce front
- Hvězdička (*) následovaná názvem skupiny správců front
- Hvězdička (*)
- Prázdný řetězec nebo řetězec obsahující všechny prázdné znaky

Jedná se o tytéž hodnoty, které lze použít pro parametr *QMgrName* v rámci volání MQCONN vydaného aplikací klienta, která používá rozhraní MQI (Message Queue Interface). Další informace o významu těchto hodnot najdete v tématu MQCONN. Vlastnost QMANAGER můžete nastavit pomocí nástroje pro administraci produktu WebSphere MQ JMS nebo pomocí Průzkumníka IBM WebSphere MQ . Alternativně může aplikace nastavit vlastnost voláním metody setQueueManager () objektu ConnectionFactory .

Pokud aplikace poté vytvoří objekt připojení z objektu ConnectionFactory , produkt IBM WebSphere MQ classes for JMS přistupuje k tabulce definic kanálů klienta identifikovanou vlastností CCDTURL, použije vlastnost QMANAGER k prohledání tabulky pro vhodnou definici kanálu připojení klienta a poté použije definici kanálu ke spuštění kanálu MQI pro správce front.

Všimněte si, že vlastnosti CCDTURL a CHANNEL objektu ConnectionFactory nemohou být nastaveny, když aplikace volá metodu createConnection(). Jsou-li nastaveny obě vlastnosti, metoda vygeneruje výjimku. Vlastnost CCDTURL nebo CHANNEL se považuje za sadu, je-li její hodnota jakákoli jiná než null, prázdný řetězec nebo řetězec obsahující všechny prázdné znaky.

Pokud produkt IBM WebSphere MQ classes for JMS najde vhodnou definici kanálu pro připojení klienta v tabulce definic kanálů klienta, použije pouze informace extrahované z tabulky ke spuštění kanálu MQI. Všechny vlastnosti související s kanálem objektu ConnectionFactory se budou ignorovat.

Zejména si všimněte následujících bodů, používáte-li zabezpečení SSL (Secure Sockets Layer):

- Kanál MQI používá zabezpečení SSL pouze v případě, že definice kanálu extrahovaná z tabulky definic kanálů klienta určuje název CipherSpec podporovaný produktem IBM WebSphere MQ classes for JMS.
- Tabulka definic kanálů klienta také obsahuje informace o umístění serverů LDAP (Lightweight Directory Access Protocol), které uchovávají seznamy zrušených certifikátů (CRL). Produkt IBM WebSphere MQ classes for JMS používá pouze tyto informace pro přístup k serverům LDAP, které obsahují seznamy CRL.

- Definiční tabulka kanálu klienta může také obsahovat umístění odpovídacího modulu OCSP (Online Certificate Status Protocol). Produkt IBM WebSphere MQ classes for JMS nemůže použít informace OCSP v souboru s tabulkou definic kanálů klienta. Nicméně můžete OCSP nakonfigurovat podle popisu uvedeného v kapitole Používání protokolu certifikátů online.

Další informace o použití SSL s tabulkou definic kanálů klienta naleznete v tématu Použití rozšířeného transakčního klienta s kanály SSL.

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 jazyce Java. To znamená například, že parametr SCYEXIT v příkazu DEFINE CHANNEL k vytvoření definice kanálu připojení klienta může určovat název třídy, která implementuje rozhraní WMQSecurityExit . Podobně může parametr SENDEXIT zadat název třídy, která implementuje rozhraní WMQSendExit , a parametr RCVEXIT může uvádět název třídy, která implementuje rozhraní WMQReceiveExit . Další informace o tom, jak zapisovat uživatelskou proceduru kanálu v prostředí Java, viz “Probíhá zápis kanálů kanálů v jazyce Java pro třídy WebSphere MQ pro JMS” na stránce 881.

  Použití uživatelských procedur kanálu napsaných v jiném jazyce, než je Java, je podporováno také. Informace o tom, jak určit parametry SCYEXIT, SENDEXIT a RCVEXIT v příkazu DEFINE CHANNEL pro uživatelské procedury kanálu zapsané v jiném jazyce, najdete v tématu DEFINE CHANNEL.

Automatické opětovné připojení klienta JMS

Konfigurujte klienta JMS, aby se znovu připojil automaticky po síti, správci front nebo selhání serveru.

Vlastnosti `CONNECTIONNAMELIST` a `CLIENTRECONNECTOPTIONS` třídy `MQConnectionFactory` se používají ke konfiguraci připojení klienta pro automatické opětovné připojení po selhání připojení nebo požadavku administrátora pro opětovné připojení klientských aplikací po zastavení správce front.

Úplný seznam názvů připojení v seznamu connectionNameje k dispozici pouze pro metody seznamugetconnectionName, které mohou pracovat se seznamem názvů připojení. Metody, jako je get/ setHostname , které nezpracovává seznamy názvů, přistupují k prvnímu jménu v seznamu.

Automaticky připojitelná klientská připojení se po navázání spojení znovu připojijí pouze po navázání spojení.

Zda aplikace pokračuje v práci správně poté, co byla znovu připojena automaticky, závisí na jejím návrhu. Přečtěte si související témata, abyste porozuměli tomu, jak navrhnout opětovné připojení klientů k databázi. Někteří existující klienti mohou pracovat správně bez úprav po automatickém opětovném navázání spojení.

Třídy WebSphere MQ pro jazyk Java automatické opětovné připojování klientů nepodporují.

Chcete-li zabránit všem klientům připojeným ke správci front, který selhal, z opětovného připojení současně, jsou pokusy o opětovné připojení zpožděny o intervaly, které jsou částečně fixní a částečně náhodné.

Při výchozím nastavení dojde k pokusům o opětovné připojení v následujících intervalech:

1. První pokus se provede po počáteční prodlevě jedné sekundy s náhodným prvkem až do 250 milisekund.
2. Druhý pokus se provede o dvě sekundy s náhodným intervalem až 500 milisekund, po selhání prvního pokusu.
3. Třetí pokus se skládá ze čtyř sekund s náhodným intervalem po jedné sekundě, po druhém pokusu o selhání.
4. Čtvrtý pokus se skládá z osmi sekund s náhodným intervalem do dvou sekund, po třetím pokusu o selhání.
5. Pátý pokus se provede 16 sekund, plus náhodný interval až 4 sekundy, po selhání čtvrtého pokusu.

6. Šestý pokus a všechny následné pokusy jsou provedeny 25 sekund s náhodným intervalem po dobu až šesti sekund a 250 milisekund po selhání předchozího pokusu.

Tento proces opakovaného připojení pokračuje, dokud se klient úspěšně znovu nepřipojí ke správci front nebo dokud neuplyne maximální interval opakovaného připojení.

Potřebujete-li zvýšit výchozí hodnoty, aby přesněji odrážely dobu potřebnou pro zotavení správce front, nebo pokud se má správce front v pohotovostním režimu aktivovat, změňte hodnoty prodlevy ve struktuře MQCLIENT.INI pomocí atributu **ReconDelay** .

Související pojmy

Automatické opětovné připojení klienta

Související úlohy

Konfigurace klienta pomocí konfiguračního souboru

Sdílení připojení TCP/IP v produktu IBM WebSphere MQ classes for JMS

Je možné vytvořit více instancí kanálu MQI, aby bylo možné sdílet jedno připojení TCP/IP.

Aplikace, které jsou spuštěny ve stejném běhovém prostředí Java a které využívají adaptér prostředků IBM WebSphere MQ classes for JMS nebo adaptér prostředků IBM WebSphere MQ pro připojení ke správci front pomocí přenosu CLIENT, lze provést tak, aby sdílely stejnou instanci kanálu.

Mezi instancemi kanálu a připojeními TCP/IP existuje vztah 1:1. Jedno připojení TCP/IP je vytvořeno pro každou instanci kanálu.

Je-li kanál definován s parametrem **SHARECNV** nastaveným na hodnotu větší než 1, pak tento počet konverzací může sdílet instanci kanálu. Chcete-li povolit továrnu připojení nebo specifikaci aktivace pro použití této funkce, nastavte vlastnost **SHARECONVALLOWED** na hodnotu YES.

Každý připojení JMS a relace JMS vytvořené aplikací JMS vytváří vlastní konverzaci se správcem front.

Po spuštění specifikace aktivace adaptér prostředků produktu IBM WebSphere MQ pro adaptér prostředků JMS zahájí konverzaci se správcem front, který má být použit pro specifikaci aktivace. Každá serverová relace ve fondu relací serveru, která je přidružena ke specifikaci aktivace, také zahájí konverzaci se správcem front.

Atribut SHARECNV je nejlepším přístupem k sdílení připojení. Proto je-li v produktu IBM WebSphere MQ classes for JMSpoužita hodnota SHARECNV větší než 0, není zaručeno, že nový požadavek na připojení bude vždy sdílet již vytvořené spojení.

Výpočet počtu instancí kanálu

Pomocí následujících vzorců určete maximální počet instancí kanálu vytvořených aplikací:

Specifikace aktivace

Počet instancí kanálu = (<**maxPoolDepth**> + 1)/<**SHARECNV**>

Kde <**maxPoolDepth**> je hodnota vlastnosti **maxPoolDepth** a <**SHARECNV**> je hodnota vlastnosti **SHARECNV** na kanálu, který je použit specifikací aktivace.

Další aplikace JMS

Počet instancí kanálu = (< připojení JMS > + < sessions>)/<**SHARECNV**>

Kde < JMS connections > je počet připojení vytvořených aplikací, kde < JMS sessions > je počet relací JMS vytvořených aplikací a <**SHARECNV**> je hodnota vlastnosti **SHARECNV** na kanálu, který je použit specifikací aktivace.

Příklady

Následující příklady ukazují způsob použití vzorců k výpočtu počtu instancí kanálu, které jsou vytvořeny ve správci front aplikacemi s použitím adaptéru prostředků IBM WebSphere MQ classes for JMS nebo IBM WebSphere MQ classes for JMS .

Příklad aplikace platformy JMS
Připojení aplikace JMS se připojuje ke správci front pomocí přenosu CLIENT a vytváří připojení JMS a tři relace JMS. Kanál, který aplikace používá pro připojení ke správci front, má vlastnost **SHARECNV** nastavenou na hodnotu 10. Je-li aplikace spuštěna, existují čtyři konverzace mezi aplikací a správcem front a jednou instancí kanálu. Všechny čtyři konverzace sdílejí instanci kanálu.

Příklad specifikace aktivace
Specifikace aktivace se připojuje ke správci front pomocí přenosu CLIENT. Specifikace aktivace je konfigurována s vlastností **maxPoolDepth** nastavenou na hodnotu 10. Kanál, který je specifikace aktivace konfigurován pro použití, má vlastnost **SHARECNV** nastavenou na hodnotu 10. Je-li specifikace aktivace spuštěna a souběžně zpracovává 10 zpráv, počet konverzací mezi specifikací aktivace a správcem front je 11 (10 konverzací pro relace serveru a jedna pro specifikaci aktivace). Počet instancí kanálu, které jsou použity specifikací aktivace, je 2.

Příklad specifikace aktivace
Specifikace aktivace se připojuje ke správci front pomocí přenosu CLIENT. Specifikace aktivace je konfigurována s vlastností **maxPoolDepth** nastavenou na hodnotu 5. Kanál, který je specifikace aktivace konfigurován pro použití, má vlastnost **SHARECNV** nastavenou na hodnotu 0. Je-li specifikace aktivace spuštěna a zpracovává se 5 zpráv souběžně, počet konverzací mezi specifikací aktivace a správcem front je 6 (pět konverzací pro relace serveru a jedna pro specifikaci aktivace). Počet instancí kanálu, které jsou použity aktivační specifikací, je 6, protože vlastnost **SHARECNV** na kanálu je nastavena na 0, každá konverzace používá vlastní instanci kanálu.

Určení rozsahu portů pro připojení klienta ve třídách produktu WebSphere MQ pro službu JMS

Použijte vlastnost LOCALADDRESS k uvedení rozsahu portů, ke kterým se vaše aplikace může vázat.

Když se třídy produktu WebSphere MQ pro aplikaci JMS pokusí připojit ke správci front WebSphere MQ v režimu klienta, může brána firewall povolit pouze připojení, která pocházejí ze zadaných portů nebo z rozsahu portů. V této situaci můžete použít vlastnost LOCALADDRESS objektu ConnectionFactory, továrny QueueConnectionFactory nebo TopicConnectionk určení portu nebo rozsahu portů, na které se aplikace může vázat.

Vlastnost LOCALADDRESS můžete nastavit pomocí nástroje pro administraci produktu WebSphere MQ JMS nebo voláním metody setLocalAddress () v aplikaci JMS. Dále je uveden příklad nastavení vlastnosti v rámci aplikace:

```
mqConnectionFactory.setLocalAddress("192.0.2.0(2000,3000)");
```

Když se aplikace připojí ke správci front následně, aplikace se připojí k lokální adrese IP a číslu portu v rozsahu 192.0.2.0(2000) na 192.0.2.0(3000).

V systému s více než jedním síťovým rozhraním můžete také použít vlastnost LOCALADDRESS k uvedení, které síťové rozhraní musí být použito pro připojení.

V případě připojení v reálném čase ke zprostředkovateli je vlastnost LOCALADDRESS relevantní pouze při použití výběrového vysílání. V takovém případě můžete použít vlastnost k určení, které lokální síťové rozhraní musí být použito pro připojení, ale hodnota vlastnosti nesmí obsahovat číslo portu nebo rozsah čísel portů.

Pokud omezíte rozsah portů, může dojít k chybám připojení. Dojde-li k chybě, je vyvolána výjimka JMSException s vloženým rozhraním MQException, které obsahuje kód příčiny WebSphere MQ MQRC_Q_MGR_NOT_AVAILABLE a následující zpráva:

```
Pokus o připojení soketu byl odmítnut kvůli omezením LOCAL_ADDRESS_PROPERTY
```

Může se vyskytnout chyba, pokud jsou použity všechny porty v uvedeném rozsahu, nebo pokud zadaná adresa IP, název hostitele nebo číslo portu nejsou platné (například záporné číslo portu).

Vzhledem k tomu, že třídy produktu WebSphere MQ pro platformu JMS mohou vytvářet jiná připojení než ta, která jsou vyžadována aplikací, vždy zvažte možnost zadání rozsahu portů. Obecně platí, že každá relace vytvořená aplikací vyžaduje jeden port a třídy produktu WebSphere MQ pro platformu JMS mohou vyžadovat tři nebo čtyři další porty. Dojde-li k chybě připojení, zvyšte rozsah portů.

Sdružování připojení, které je používáno při výchozím nastavení ve třídách WebSphere MQ pro službu JMS, může mít vliv na rychlost, kterou lze znovu použít porty. V důsledku toho může dojít k chybě připojení, zatímco jsou porty uvolněny.

Komprese kanálu ve třídách produktu WebSphere MQ pro službu JMS

Třídy WebSphere MQ pro aplikaci JMS mohou používat zařízení produktu WebSphere MQ ke komprimování záhlaví nebo dat zprávy.

Komprimace dat, která teče na kanálu produktu WebSphere MQ , může zlepšit výkon kanálu a omezit provoz na síti. Pomocí funkce dodávané s produktem WebSphere MQje možné komprimovat data, která proudí na kanály zpráv a kanály MQI. U obou typů kanálů můžete komprimovat data záhlaví a data zprávy nezávisle na sobě. Standardně nejsou žádná data komprimována na kanálu.

Třídy WebSphere MQ pro aplikaci JMS určují metody, které lze použít pro kompresi dat záhlaví nebo zprávy na připojení vytvořením objektu java.util.Collection . Každá kompresní technika je objekt Integer v kolekci a pořadí, ve kterém aplikace přidá techniky komprese do kolekce, je pořadí, ve kterém jsou techniky komprese vyjednané se správcem front, když aplikace vytvoří připojení. Aplikace pak může předat kolekci do objektu ConnectionFactory voláním metody setHdrCompList(), pro data záhlaví nebo metodou setMsgCompList() pro data zprávy. Je-li aplikace připravena, může vytvořit připojení.

Následující fragmenty kódu popisují popsaný přístup. První fragment kódu ukazuje, jak implementovat kompresi dat záhlaví:

```
Collection headerComp = new Vector();
headerComp.add(new Integer(WMQConstants.WMQ_COMPHDR_SYSTEM));
.
.
.
((MQConnectionFactory) cf).setHdrCompList(headerComp);
.
.
.
connection = cf.createConnection();
```

Druhý fragment kódu ukazuje, jak implementovat kompresi dat zprávy:

```
Collection msgComp = new Vector();
msgComp.add(new Integer(WMQConstants.WMQ_COMPMSG_RLE));
msgComp.add(new Integer(WMQConstants.WMQ_COMPMSG_ZLIBHIGH));
.
.
.
((MQConnectionFactory) cf).setMsgCompList(msgComp);
.
.
.
connection = cf.createConnection();
```

Ve druhém příkladu jsou techniky komprese vyjednané v pořadí RLE, pak ZLIBHIGH, když se vytvoří připojení. Zvolená technika komprese nemůže být změněna během doby životnosti objektu připojení. Chcete-li pro připojení použít kompresi, musí být před vytvořením objektu Connection volány metody setHdrCompList() a setMsgCompList().

Asynchronně vkládání zpráv do tříd produktu IBM WebSphere MQ pro platformu JMS

Obvykle, když aplikace odesílá zprávy do místa určení, musí aplikace čekat, až správce front potvrdí, že zpracoval požadavek. Můžete zlepšit výkon systému zpráv za určitých okolností tím, že zvolíte, že nebudete asynchronně vkládat zprávy. Když aplikace asynchronně odešle zprávu, správce front nevrátí úspěch nebo selhání každého volání, ale můžete namísto toho zkontrolovat chyby pravidelně.

Určuje, zda místo určení vrátí řízení aplikaci, aniž by určujete, zda správce front zprávu přijal bezpečně, závisí na následujících vlastnostech:

- Cílová vlastnost JMS PUTSYNCALLOWED (krátký název-PAALD).

  Parametr PUTASYNCALLOWED řídí, zda aplikace platformy JMS mohou asynchronně vkládat zprávy, je-li tato volba povolena pro základní frontu nebo téma, které cíl JMS reprezentuje.

- Vlastnost fronty IBM WebSphere MQ nebo tématu DEFPRESP (Výchozí typ odezvy put).

  DEFPRESP určuje, zda aplikace, které vložila zprávy do fronty nebo publikují zprávy do daného tématu, mohou využívat funkčnost asynchronního vložení.

Následující tabulka uvádí možné hodnoty vlastností PUTASYNCALLOWED a DEFPRESP a hodnoty, které jsou vyžadovány pro funkčnost asynchronního vkládání, jsou-li povoleny:

Tabulka 131. Vlastnosti PUTASYNCALLOWED a DEFPRESP určují, zda jsou zprávy vsazeny asynchronně.

Vlastnost fronty produktu WebSphere MQ	**PUTASYNCALLOWED = NE**	**PUTASYNCALLOWED = ANO**	**PUTASYNCALLOWED = AS_DEST nebo AS_Q_DEF nebo AS_T_DEF**
DEFPRESP=SYNCHRONIZACE	Funkčnost asynchronního vložení není povolena	Povolena funkčnost asynchronního vkládání	Funkčnost asynchronního vložení není povolena
DEFPRESP=ASYNC	Funkčnost asynchronního vložení není povolena	Povolena funkčnost asynchronního vkládání	Povolena funkčnost asynchronního vkládání

Pro zprávy odeslané v relaci transakce nakonec aplikace určí, zda správce front přijal zprávy bezpečně při volání produktu `commit()`.

Pokud aplikace odesílá trvalé zprávy v rámci relace s transakcemi a jedna nebo více zpráv není přijato bezpečně, transakce se nezdaří a vygeneruje výjimku. Pokud však aplikace odešle přechodné zprávy v rámci relace s transakcemi a jedna nebo více zpráv není bezpečně doručeno, transakce se úspěšně potvrdí. Aplikace neobdrží žádnou zpětnou vazbu, že přechodné zprávy nedorazly bezpečně.

Pro přechodné zprávy odeslané v relaci, která není součástí transakce, určuje vlastnost SENDCHECKCOUNT objektu *ConnectionFactory* , kolik zpráv má být odesláno, před třídami produktu IBM WebSphere MQ pro službu JMS, které správce front úspěšně přijal zprávy.

Pokud kontrola zjistí, že jedna nebo více zpráv nebyla bezpečně přijata a aplikace zaregistrovala modul listener pro výjimky s připojením, třídy IBM WebSphere MQ pro platformu JMS volají metodu `onException()` modulu listener výjimek k předání výjimky JMS do aplikace.

Výjimka JMS má kód chyby JMSWMQ0028 a tento kód zobrazí následující zprávu:

```
At least one asynchronous put message failed or gave a warning.
```

Výjimka JMS má také připojenou výjimku, která poskytuje více podrobností. Výchozí hodnota vlastnosti SENDCHECKCOUNT je nula, což znamená, že žádné takové kontroly nejsou provedeny.

Tato optimalizace má největší přínos pro aplikaci, která se připojuje ke správci front v režimu klienta, a potřebuje odeslat posloupnost zpráv v rychlém sledu, ale nevyžaduje okamžitou zpětnou vazbu od správce front pro každou odeslanou zprávu. Aplikace však přesto může tuto optimalizaci použít i v případě, že se připojuje ke správci front v režimu vazeb, ale očekávaný přínos výkonu není tak velký.

Použití dopředného čtení s třídami produktu WebSphere MQ pro službu JMS

Funkce dopředného čtení, kterou poskytuje produkt WebSphere MQ umožňuje netrvalé zprávy, které jsou přijaty mimo transakci, odeslány do produktu IBM WebSphere MQ classes for JMS před tím, než je aplikace vyžádá. Server IBM WebSphere MQ classes for JMS ukládá zprávy ve vnitřní vyrovnávací paměti a předává zprávy aplikaci, když se o ně aplikace požádá.

Aplikace produktu IBM WebSphere MQ classes for JMS , které používají produkt `MessageConsumers` nebo `MessageListeners` k příjmu zpráv z místa určení mimo transakci, mohou využívat funkce čtení napřed. Použití dopředného čtení umožňuje aplikacím, které tyto objekty využívají, aby mohly těžit ze zlepšení výkonu při příjmu zpráv.

To, zda aplikace, která používá produkt `MessageConsumers` nebo `MessageListeners` , může dopředné čtení použít, závisí na následujících vlastnostech:

- Cílová vlastnost JMS READAHEADALLOWED (krátký název-RAALD). READAHEADALLOWED řídí, zda aplikace platformy JMS mohou při získávání nebo procházení dočasných zpráv mimo transakci používat dopředné čtení, pokud tato volba povoluje základní frontu nebo téma, které cíl JMS reprezentuje.

- Vlastnost fronty IBM WebSphere MQ nebo tématu DEFREADA (Výchozí dopředné čtení). DEFREADA určuje, zda aplikace, které přijímají nebo prohledávaní dočasné zprávy mimo transakci, mohou použít dopředné čtení.

Následující tabulka uvádí možné hodnoty vlastností READAHEADALLOWED a DEFREADA a jaké hodnoty jsou vyžadovány pro funkce čtení napřed, které mají být povoleny:

Tabulka 132. Vlastnosti READAHEADALLOWED a DEFREADA určuje, zda je při příjmu nebo procházení dočasných zpráv mimo transakci použito dopředné čtení.

Cílová vlastnost produktu WebSphere MQ	**READAHEADALLOWED = ANO**	**READAHEADALLOWED = NO**	**AS_DEST nebo AS_Q_DEF nebo AS T_DEF**
Vlastnost fronty produktu WebSphere MQ			
DEFREADA = NE	Povolena funkčnost čtení napřed	Funkce dopředného čtení není povolena	Funkce dopředného čtení není povolena
DEFREADA = ANO	Povolena funkčnost čtení napřed	Funkce dopředného čtení není povolena	Povolena funkčnost čtení napřed
DEFREADA = VYPNUTO	Funkce dopředného čtení není povolena	Funkce dopředného čtení není povolena	Funkce dopředného čtení není povolena

Je-li povolena funkce dopředného čtení, je-li `MessageConsumer` nebo `MessageListener` vytvořena aplikací, IBM WebSphere MQ classes for JMS vytvoří vnitřní vyrovnávací paměť pro cíl, který je monitorován produktem `MessageConsumer` nebo `MessageListener` . Pro každý `MessageConsumer` nebo `MessageListener`existuje jedna vnitřní vyrovnávací paměť. Správce front začne odesílat netrvalé zprávy do serveru IBM WebSphere MQ classes for JMS , když aplikace volá jednu z následujících metod:

- `MessageConsumer.receive()`
- `MessageConsumer.receive(long timeout)`
- `MessageConsumer.receiveNoWait()`
- `Session.setMessageListener(MessageListener listener)`

Produkt IBM WebSphere MQ classes for JMS automaticky vrátí první zprávu zpět do aplikace, metodou volání metody, kterou provedla aplikace. Ostatní netrvalé zprávy jsou uloženy produktem IBM WebSphere MQ classes for JMS ve vnitřní vyrovnávací paměti, která byla vytvořena pro místo určení. Když aplikace požádá o další zprávu ke zpracování, vrátí IBM WebSphere MQ classes for JMS další zprávu ve vnitřní vyrovnávací paměti.

IBM WebSphere MQ classes for JMS požaduje od správce front další přechodné zprávy, je-li vnitřní vyrovnávací paměť prázdná.

Vnitřní vyrovnávací paměť, kterou používá IBM WebSphere MQ classes for JMS , se odstraní, když aplikace uzavře `MessageConsumer`nebo relaci JMS, se kterou je `MessageListener` přidružen.

Pro systém `MessageConsumers`se všechny nezpracované zprávy v interní vyrovnávací paměti ztratí.

Při použití produktu `MessageListeners`, to, co se stane s zprávami v interní vyrovnávací paměti, závisí na vlastnosti cíle JMS READAHEADCLOSEPOLICY (krátký název-RACP). Výchozí hodnota vlastnosti je DELIVER_ALL, což znamená, že relace JMS, která byla použita k vytvoření `MessageListener` , není zavřena, dokud se do aplikace nedoručí všechny zprávy ve vnitřní vyrovnávací paměti. Je-li vlastnost nastavena na DELIVER_CURRENT, bude relace JMS zavřena poté, co byla aktuální zpráva zpracována aplikací a všechny zbývající zprávy v interní vyrovnávací paměti budou zrušeny.

Zachovaná publikování v třídách WebSphere MQ pro JMS

Třídy WebSphere MQ pro klienta JMS mohou být konfigurovány k použití zachovaných publikování.

Vydavatel může určit, že kopie publikace musí být uchována tak, aby mohla být odeslána na budoucí odběratele, kteří se zajímají o dané téma. Toto se provádí ve třídách produktu WebSphere MQ pro platformu JMS nastavením celočíselné vlastnosti JMS_IBM_RETAIN na hodnotu 1. Pro tyto hodnoty byly definovány konstanty v rozhraní com.ibm.msg.client.jms.JmsConstants . Pokud jste například vytvořili zprávu *msg*a nastavili ji jako zachované publikování, použijte následující kód:

```
// set as a retained publication
msg.setIntProperty(JmsConstants.JMS_IBM_RETAIN, JmsConstants.RETAIN_PUBLICATION);
```

Nyní můžete odeslat zprávu jako normální. V přijaté zprávě lze obdržet dotaz na službu JMS_IBM_RETAIN. Je tedy možné se dotázat, zda je přijatá zpráva zachovaná publikace.

Podpora standardu XA v třídách WebSphere MQ pro službu JMS

Platforma JMS podporuje transakce kompatibilní s podporou XA ve vazbách a v režimu klienta s podporovaným správcem transakcí.

Pokud v prostředí aplikačního serveru vyžadujete funkčnost XA, je třeba aplikaci odpovídajícím způsobem nakonfigurovat. Informace o tom, jak konfigurovat aplikace k použití distribuovaných transakcí, naleznete v dokumentaci k aplikačnímu serveru.

Použití připojení v reálném čase k zprostředkovateli produktu WebSphere Event Broker nebo zprostředkovatele zpráv WebSphere Message Broker

Třídy WebSphere MQ pro aplikaci JMS mohou používat v reálném čase připojení ke zprostředkovateli produktu WebSphere Event Broker nebo zprostředkovatele zpráv WebSphere Message Broker pro systém zpráv publikování/odběru. Jak zprostředkovatel, tak i třídy WebSphere MQ pro rozhraní JMS musí být nakonfigurovány tak, aby povolovali připojení v reálném čase.

Když aplikace používá v reálném čase připojení ke zprostředkovateli produktu WebSphere Event Broker nebo zprostředkovatele zpráv WebSphere Message Broker, aplikace a zprostředkovatel výměnu zpráv pomocí produktu WebSphere MQ Real-Time Transport. V závislosti na konfiguraci mohou být zprávy doručovány do aplikace pomocí přenosu výběrového vysílání WebSphere MQ .

Informace o tom, jak se aplikace může připojit ke správci front produktu WebSphere MQ a používat produkt WebSphere MQ Enterprise Transport k výměně zpráv se zprostředkovatelem produktu WebSphere Event Broker nebo WebSphere Message Broker, naleznete v dokumentaci k předchozím verzím tříd produktu WebSphere MQ pro platformu JMS. Nezapomeňte, že při použití produktu WebSphere MQ Enterprise Transport se musí aplikace připojit ke správci front s použitím továrny připojení, která je spuštěna v režimu migrace poskytovatele systému zpráv WebSphere MQ .

Konfigurace zprostředkovatele produktu WebSphere Event Broker nebo zprostředkovatele zpráv WebSphere Message Broker pro připojení v reálném čase

V případě aplikací WebSphere MQ pro aplikaci JMS pro použití v reálném čase pro zprostředkovatele produktu WebSphere Event Broker nebo zprostředkovatele zpráv WebSphere Message Broker je třeba zprostředkovatele nakonfigurovat tak, že vytvoříte a implementujete tok zpráv pro čtení zpráv z portu TCP/IP, na kterém zprostředkovatel přijímá zprávy a publikuje zprávy. V závislosti na vašich požadavcích budete možná muset zprostředkovatele nakonfigurovat dalšími způsoby.

Chcete-li konfigurovat zprostředkovatele, musíte vytvořit a implementovat jeden z následujících toků zpráv:

- Tok zpráv, který obsahuje uzel zpracování zpráv toku Real-timeOptimized.
- Tok zpráv, který obsahuje uzel zpracování zpráv Real-timeInput a uzel zpracování zpráv publikování.

Chcete-li naslouchat na portu TCP/IP používaném pro připojení v reálném čase, musíte nakonfigurovat uzel Real-timeOptimizedFlow nebo Real-timeInput . Ve výchozím nastavení je číslo portu pro připojení v reálném čase 1506.

Zprostředkovatele je třeba konfigurovat také v případě, že máte některý z následujících požadavků:

- Chcete-li, aby se aplikace připojovala k zprostředkovateli pomocí ověření protokolu SSL (Secure Sockets Layer),
- Chcete-li, aby se aplikace připojovala ke zprostředkovateli pomocí směrování HTTP Tunnelling,
- Pokud chcete, aby zprávy byly doručovány spotřebiteli zpráv pomocí výběrového vysílání

Informace o tom, jak konfigurovat zprostředkovatele, viz *WebSphere Dokumentace k produktu zprostředkovatele událostí* nebo *WebSphere Dokumentace k produktu Message Broker*.

Konfigurace tříd produktu WebSphere MQ pro službu JMS pro připojení v reálném čase ke zprostředkovateli produktu WebSphere Event Broker nebo zprostředkovatele zpráv WebSphere Message Broker

U tříd produktu WebSphere MQ pro aplikaci JMS pro použití připojení v reálném čase ke zprostředkovateli produktu WebSphere Event Broker nebo WebSphere Message Broker musí být třídy WebSphere MQ pro systém JMS konfigurovány nastavením určitých vlastností faktorie připojení. V závislosti na vašich požadavcích může být třeba nakonfigurovat třídy produktu WebSphere MQ pro platformu JMS dalšími způsoby.

Chcete-li konfigurovat třídy WebSphere MQ pro službu JMS, musí být nastaveny následující vlastnosti továrny připojení:

- Vlastnost TRANSPORT musí být nastavena na hodnotu DIRECT.

  Aby se však aplikace mohla připojit pomocí tunelování HTTP, musí být místo toho vlastnost TRANSPORT nastavena na hodnotu REHS CTHTTP. Viz “Použití tunelového propojení HTTP” na stránce 894.
- Vlastnost HOSTNAME musí být nastavena na název hostitele nebo adresu IP systému, na kterém je zprostředkovatel spuštěn.
- Vlastnost PORT musí být nastavena na číslo portu, na kterém zprostředkovatel naslouchá pro připojení v reálném čase.

Aplikace může tyto vlastnosti nastavit dynamicky za běhu pomocí rozšíření JMS IBM nebo rozšíření WebSphere MQ JMS. Případně, je-li továrna připojení administrovaný objekt, může administrátor nastavit tyto vlastnosti pomocí nástroje pro administraci produktu WebSphere MQ JMS nebo Průzkumníka WebSphere MQ .

Informace o vlastnostech a metodách používaných aplikacemi k nastavení jejich hodnot naleznete v tématu Vlastnosti objektů IBM WebSphere MQ classes for JMS. Informace o způsobu použití nástroje pro administraci produktu WebSphere MQ JMS naleznete v příručce “Použití nástroje pro administraci produktu WebSphere MQ JMS” na stránce 902. Informace o tom, jak používat produkt WebSphere MQ Explorer, naleznete v nápovědě k produktu WebSphere MQ Explorer.

Máte-li některý z následujících požadavků, třídy WebSphere MQ pro JMS vyžadují další konfiguraci:

- Chcete-li, aby se aplikace připojovala ke zprostředkovateli pomocí ověřování pomocí protokolu SSL (Secure Sockets Layer),
- Chcete-li, aby se aplikace připojovala ke zprostředkovateli pomocí směrování HTTP Tunnelling,
- Chcete-li aplikaci připojit k danému zprostředkovateli prostřednictvím serveru proxy,
- Pokud chcete, aby zprávy byly doručovány spotřebiteli zpráv pomocí výběrového vysílání

Následující části popisují postup konfigurace tříd produktu WebSphere MQ pro platformu JMS pro každý z těchto požadavků.

Použití ověření Secure Sockets Layer (SSL)

Ověřování SSL lze použít v reálném čase připojení k danému zprostředkovateli. Pro tento typ připojení je podporováno pouze ověření. Nemůžete použít SSL k šifrování a dešifrování dat zpráv, která teče mezi aplikací a brokerem, nebo k detekci falšování dat.

Všimněte si rozdílu mezi touto situací a tím, že když se aplikace připojí ke správci front v režimu klienta. V druhém případě můžete použít podporu zabezpečení SSL produktu WebSphere MQ k šifrování

a dešifrování dat zpráv, která proudí mezi aplikací a správcem front, a k detekci falšování dat a k poskytnutí ověření.

Chcete-li chránit data zpráv v reálném čase připojení k zprostředkovateli, můžete místo toho použít funkci poskytovanou zprostředkovatelem. Ke každému tématu se zprávami, které chcete chránit, můžete přiřadit hodnotu kvality ochrany (QoP). Pro každé téma proto můžete vybrat jinou úroveň ochrany zpráv. Další informace o ochraně zpráv poskytované zprostředkovatelem naleznete v příručce *WebSphere Dokumentace k produktu zprostředkovatele událostí* nebo *WebSphere Dokumentace k produktu Message Broker*.

Chcete-li používat ověřování SSL v reálném čase připojení k zprostředkovateli, vlastnost DIRECTAUTH továrny připojení musí být nastavena na hodnotu CERTIFICATE.

Pokud chcete použít SSL pro vzájemné ověření, vlastnost Typ protokolu ověření zprostředkovatele musí uvádět volbu R pro symetrické SSL. Chcete-li použít zabezpečení SSL pouze pro ověření zprostředkovatele, vlastnost Typ protokolu ověření zprostředkovatele musí uvádět volbu S pro asymetrické zabezpečení SSL. V takovém případě se však musí aplikace připojit ke zprostředkovateli voláním metody createConnection() s ID uživatele a heslem jako parametry, jako v následujícím příkladu:

```
factory.createConnection("user1", "user1pw");
```

Zprostředkovatel pak pro ověření aplikace použije ID uživatele a heslo místo SSL. Další informace o tom, jak konfigurovat zprostředkovatele pro ověření SSL viz *WebSphere Dokumentace k produktu zprostředkovatele událostí* nebo *WebSphere Dokumentace k produktu Message Broker*.

Notes:

1. Hodnota vlastnosti DIRECTAUTH určuje, zda je ověřování SSL používáno v reálném čase pro připojení k zprostředkovateli, nikoli k hodnotě vlastnosti SSLCIPHERSUITE.
2. Je-li ověřování SSL používáno v reálném čase připojení k danému zprostředkovateli, použijí se vlastnosti SSLPEERNAME a SSLCRL k provádění stejných kontrol jako ty, které jsou prováděny při připojování aplikace ke správci front v režimu klienta.
3. Třídy WebSphere MQ pro platformu JMS mohou používat stejné úložiště klíčů a úložiště údajů o důvěryhodnosti rozšíření JSSE (Java Secure Socket Extension), aby poskytovaly podporu zabezpečení SSL v následujících situacích:
   - Když aplikace používá v reálném čase připojení ke zprostředkovateli
   - Když se aplikace připojí ke správci front v režimu klienta

Použití tunelového propojení HTTP

Třídy WebSphere MQ pro aplikaci JMS se mohou připojit ke zprostředkovateli pomocí HTTP Tunnelling, což znamená, že se aplikace připojuje ke zprostředkovateli pomocí protokolu HTTP, jako by se připojoval k webu.

Chcete-li použít tunelování HTTP v reálném čase připojení k zprostředkovateli, musí být vlastnost TRANSPORT továrny připojení nastavena na hodnotu DIRECTHTTP.

Tunelování HTTP nelze použít ve spojení s ověřením SSL, připojením přes proxy server nebo doručením zpráv pomocí výběrového vysílání. Podporovaná verze protokolu HTTP je 1.0. Verze HTTP 1.1 není podporována.

Připojení prostřednictvím serveru proxy

Třídy WebSphere MQ pro aplikaci JMS mohou používat v reálném čase připojení ke zprostředkovateli prostřednictvím připojení k serveru proxy. Třídy WebSphere MQ pro rozhraní JMS se připojují přímo k serveru proxy a používají internetový protokol definovaný v RFC 2817, aby požádal server proxy o postoupení žádosti o připojení zprostředkovateli.

Chcete-li se připojit k zprostředkovateli prostřednictvím serveru proxy, musí být nastaveny následující vlastnosti továrny připojení:

- Vlastnost PROXYHOSTNAME musí být nastavena na název hostitele nebo adresu IP systému, na kterém je spuštěn server proxy.
- Vlastnost PROXYPORT musí být nastavena na číslo portu, na kterém naslouchá server proxy.

Není-li vlastnost PROXYHOSTNAME nastavena nebo je nastavena na prázdný řetězec, třídy WebSphere MQ pro rozhraní JMS se pokusí o připojení přímo ke zprostředkovateli pouze pomocí vlastností HOSTNAME a PORT a nepokusí se o připojení přes server proxy.

Dodání zpráv pomocí výběrového vysílání

Při použití připojení v reálném čase ke zprostředkovateli lze zprávy doručovat spotřebiteli zpráv pomocí výběrového vysílání.

Chcete-li povolit výběrové vysílání, musí být vlastnost MULTICAST objektu Topic nastavena na požadovanou volbu výběrového vysílání. Je-li vlastnost MULTICAST objektu Topic nastavena na hodnotu ASCF, musí být vlastnost MULTICAST továrny připojení nastavena na požadovanou volbu výběrového vysílání.

Třídy WebSphere MQ pro platformu JMS podporují protokoly PPL (Packet Transfer Layer) i multicast (PGM) Multicast, a zahrnuje podporu pro obě implementace protokolu PGM, PGM/IP a PGM UDP zapouzdřené. Podpora PGM/IP je však k dispozici pouze na následujících platformách:

- AIX (pouze 32 bitů)
- Linux (platformax86)
- Linux (platforma zSeries , 32bitová pouze)
- Solaris SPARC (pouze 32 bitů)
- Windows (pouze 32 bitů)
- z/OS

Třídy produktu WebSphere MQ pro zařízení aplikačního serveru JMS

Toto téma popisuje, jak produkt WebSphere MQ Classes for JMS implementuje třídu ConnectionConsumer a rozšířenou funkčnost ve třídě relace. Shrnuje také souhrn funkce fondu relací serveru.

Třídy WebSphere MQ pro platformu JMS podporují funkce ASF (Application Server Facilities), které jsou určeny ve specifikaci *Java Message Service Specification, verze 1.1* (viz webový server Java společnosti Sun na adrese https://java.sun.com). Tato specifikace identifikuje tři role v rámci tohoto programovacího modelu:

- **Poskytovatel rozhraní JMS** poskytuje funkce ConnectionConsumer a rozšířené funkce relace.
- **Aplikační server** poskytuje funkce ServerSessionPool a ServerSession .
- **Aplikace typu klient** používá funkce, které poskytovatel JMS a dodávka aplikačního serveru.

Informace v tomto tématu se nepoužijí, pokud aplikace používá v reálném čase připojení ke zprostředkovateli.

Služba JMS ConnectionConsumer

Rozhraní ConnectionConsumer poskytuje výkonnou metodu k souběžnému doručování zpráv do fondu podprocesů.

Specifikace JMS umožňuje aplikačnímu serveru integraci úzce s implementací platformy JMS pomocí rozhraní produktu `ConnectionConsumer` . Tato funkce poskytuje souběžné zpracování zpráv. Aplikační server obvykle vytvoří fond podprocesů a implementace platformy JMS zpřístupní tyto zprávy těmto podprocesům. Aplikační server s podporou JMS (například WebSphere Application Server) může tuto funkci použít k zajištění funkčnosti systému zpráv na vysoké úrovni, jako jsou objekty bean řízené zprávami.

Běžné aplikace nepoužívají ConnectionConsumer, ale mohou ji používat expertní klienti JMS. Pro takové klienty nabízí ConnectionConsumer výkonnou metodu k souběžnému doručování zpráv do fondu

podprocesů. Když zpráva dorazí do fronty nebo tématu, platforma JMS vybere podproces z fondu a doručí do ní dávku zpráv. Chcete-li toto provést, systém JMS spustí přidruženou metodu onMessage() metody MessageListener.

Stejného efektu lze dosáhnout vytvořením více objektů Session a MessageConsumer , přičemž každý má registrovaný objekt MessageListener. Hodnota ConnectionConsumer však poskytuje lepší výkon, menší využití prostředků a větší flexibilitu. Požaduje se zejména, že je potřeba méně objektů relace.

Plánování aplikace pomocí ASF

Tento oddíl popisuje, jak naplánovat aplikaci zahrnující:

- "Obecné zásady pro výměnu zpráv mezi dvěma body pomocí ASF" na stránce 896
- "Obecné zásady pro publikování/odběr systému zpráv pomocí ASF" na stránce 897
- "Odebrání zpráv z fronty v ASF" na stránce 897
- Ošetřování škodlivých zpráv v ASF. Viz "Zpracování nezpracovatelných zpráv v produktu IBM WebSphere MQ classes for JMS" na stránce 859.

Obecné zásady pro výměnu zpráv mezi dvěma body pomocí ASF

Toto téma obsahuje obecné informace o systému zpráv typu point-to-point s použitím ASF.

Když aplikace vytvoří objekt ConnectionConsumer z objektu QueueConnection , určuje objekt fronty JMS a řetězec selektoru. Volba ConnectionConsumer poté začne poskytovat zprávy pro relace v přidružené oblasti ServerSession. Zprávy dorazí do fronty a pokud se shodují s selektorem, jsou doručeny do relací v přidružené oblasti ServerSession.

V produktu WebSphere MQ odkazuje objekt fronty na QLOCAL nebo QALIAS na lokálním správci front. Je-li to QALIAS, musí se QALIAS odkazovat na QLOCAL. Úplný vyřešený produkt WebSphere MQ QLOCAL je známý jako *základní QLOCAL*. ConnectionConsumer má být *aktivní* , pokud není zavřen a jeho nadřízený objekt QueueConnection je spuštěn.

Je možné, aby pro více ConnectionConsumers, každá s různými selektory, bylo spuštěno na stejném podkladovém prostředí QLOCAL. Chcete-li zachovat výkonnost, nesmí se nežádoucí zprávy hromadit ve frontě. Nechtěné zprávy jsou ty, pro které nemá žádný aktivní ConnectionConsumer odpovídající selektor. Továrnu QueueConnectionmůžete nastavit tak, aby byly tyto nežádoucí zprávy odebrány z fronty (podrobnosti viz "Odebrání zpráv z fronty v ASF" na stránce 897). Toto chování můžete nastavit jedním ze dvou způsobů:

- Pomocí nástroje pro administraci platformy JMS nastavte továrnu QueueConnectionna hodnotu MRET (NO).
- Ve vašem programu použijte:

```
MQQueueConnectionFactory.setMessageRetention(WMQConstants.WMQ_MRET_NO)
```

Pokud toto nastavení nezměníte, bude pro výchozí nastavení zachovány tyto nežádoucí zprávy ve frontě.

Při nastavení správce front produktu WebSphere MQ vezměte v úvahu následující body:

- Základní hodnota QLOCAL musí být povolena pro sdílený vstup. Chcete-li to provést, použijte následující příkaz MQSC:

```
ALTER QLOCAL(your.qlocal.name) SHARE GET(ENABLED)
```

- Váš správce front musí mít povolenu frontu nedoručených zpráv. Pokud má vlastnost ConnectionConsumer problém, když umístí zprávu do fronty nedoručených zpráv, doručení zprávy ze základní fronty QLOCAL se zastaví. Chcete-li definovat frontu nedoručených zpráv, použijte:

```
ALTER QMGR DEADQ(your.dead.letter.queue.name)
```

- Uživatel, který spouští objekt ConnectionConsumer , musí mít oprávnění k provedení operace MQOPEN s MQOO_SAVE_ALL_CONTEXT a MQOO_PASS_ALL_CONTEXT. Další informace naleznete v dokumentaci produktu WebSphere MQ pro konkrétní platformu.
- Pokud jsou nežádoucí zprávy ponechány ve frontě, sníží se výkon systému. Proto plánujte své selektory zpráv tak, aby se mezi nimi ConnectionConsumers odebrali všechny zprávy z fronty.

Podrobnosti o příkazech MQSC naleznete v příručce Odkaz na MQSC .

Obecné zásady pro publikování/odběr systému zpráv pomocí ASF

Volba ConnectionConsumers přijímá zprávy pro určené téma. Objekt ConnectionConsumer může být trvalý nebo dočasný. Musíte určit, kterou frontu nebo fronty má ConnectionConsumer používat.

Když aplikace vytvoří objekt ConnectionConsumer z objektu TopicConnection , určuje objekt Topic a řetězec selektoru. Hodnota ConnectionConsumer poté začne přijímat zprávy, které odpovídají selektoru na daném tématu, včetně všech zachovaných publikování pro odebírané téma.

Volitelně může aplikace vytvořit trvalý ConnectionConsumer , který je přidružen ke specifickému názvu. Tento ConnectionConsumer přijímá zprávy, které byly publikovány na téma od poslední aktivace trvanlivého ConnectionConsumer . Obdrží všechny takové zprávy, které se shodují se selektorem na tématu. Pokud však hodnota ConnectionConsumer používá dopředné čtení, může ztratit přechodné zprávy, které se nacházejí ve vyrovnávací paměti klienta při zavření.

Je-li třída produktu WebSphere MQ pro službu JMS v režimu migrace poskytovatele systému zpráv produktu WebSphere MQ , je pro netrvalé odběry ConnectionConsumer použita samostatná fronta.Konfigurovatelná volba CCSUB u továrny TopicConnectionurčuje frontu, která má být použita. Obvykle příkaz CCSUB určuje jednu frontu pro použití všemi prvky ConnectionConsumers , které používají stejné Továrny TopicConnection. Je však možné, aby každý ConnectionConsumer generoval dočasnou frontu uvedením prefixu názvu fronty následovaného hvězdičkou (*).

Pokud jsou třídy produktu WebSphere MQ pro službu JMS v režimu migrace poskytovatele systému zpráv produktu WebSphere MQ , vlastnost CCDSUB daného tématu určuje frontu, která má být použita pro trvalé odběry. Opět platí, že se jedná o frontu, která již existuje, nebo předponu názvu fronty, za kterou následuje hvězdička (*). Uvedete-li frontu, která již existuje, budou všechny trvalé ConnectionConsumers , které se přihlašují k tématu, používají tuto frontu. Uvedete-li předponu názvu fronty následovanou hvězdičkou (*), vygeneruje se fronta poprvé, kdy se vytvoří trvalý ConnectionConsumer s konkrétním názvem. Tato fronta je znovu použita později, je-li vytvořen trvalý objekt ConnectionConsumer se stejným názvem.

Při nastavení správce front produktu WebSphere MQ vezměte v úvahu následující body:

- Váš správce front musí mít povolenu frontu nedoručených zpráv. Pokud má vlastnost ConnectionConsumer problém, když umístí zprávu do fronty nedoručených zpráv, doručení zprávy ze základní fronty QLOCAL se zastaví. Chcete-li definovat frontu nedoručených zpráv, použijte:

```
ALTER QMGR DEADQ(your.dead.letter.queue.name)
```

- Uživatel, který spouští objekt ConnectionConsumer , musí mít oprávnění k provedení operace MQOPEN s MQOO_SAVE_ALL_CONTEXT a MQOO_PASS_ALL_CONTEXT. Další informace naleznete v dokumentaci produktu WebSphere MQ pro příslušnou platformu.
- Výkon pro jednotlivé ConnectionConsumer můžete optimalizovat vytvořením samostatné, vyhrazené, fronty pro tuto frontu. Jedná se o náklady na využití dodatečných prostředků.

Odebrání zpráv z fronty v ASF

Když aplikace používá ConnectionConsumers, může služba JMS odebírat zprávy z fronty v řadě situací.

Tyto situace jsou následující:

Chybně formátovaná zpráva
Může dojít k tomu, že zpráva JMS nemůže analyzovat.

Nezpracovatelná zpráva
Zpráva se může dostat do prahové hodnoty vrácení, ale ConnectionConsumer ji neopětuje ve frontě vrácení.

Není zájem ConnectionConsumer
Pro systém zpráv typu point-to-point, je-li továrna QueueConnectionnastavena tak, aby neuchovávaly nežádoucí zprávy, je doručena zpráva, která je nechtěná některým z ConnectionConsumers.

V takových situacích se ConnectionConsumer pokusí o odebrání zprávy z fronty. Volby odebrání v poli sestavy v rámci zprávy MQMD zprávy nastavují přesné chování. Tyto volby jsou:

MQRO_DEAD_LETTER_Q
Zpráva je znovu zařazena do fronty nedoručených zpráv správce front. Toto nastavení je výchozí.

MQRO_DISCARD_MSG
Zpráva byla zrušena.

Volba ConnectionConsumer také vygeneruje zprávu sestavy a to také závisí na poli sestavy MQMD zprávy. Tato zpráva se odešle na zprávu ReplyToQ zprávy na ReplyToQmgr. Pokud dojde k chybě při odesílání zprávy sestavy, bude místo toho odeslána zpráva do fronty nedoručených zpráv. Volby sestavy výjimek v poli sestavy v podrobnostech sestavy MQMD zprávy sestavy. Tyto volby jsou:

VÝJIMKA MQRO_EXCEPTION
Bude vygenerována zpráva sestavy obsahující MQMD původní zprávy. Neobsahuje žádná těla zprávy.

MQRO_EXCEPTION_WIT_DATA
Bude vygenerována zpráva sestavy obsahující deskriptor MQMD, všechny záhlaví MQ a 100 bajtů dat těla.

MQRO_EXCEPTION_WITH_FULL_DATA
Bude vygenerována zpráva sestavy, která obsahuje všechna data z původní zprávy.

default
Negeneruje se žádná zpráva sestavy.

Když se vygenerují zprávy sestav, jsou uznány následující volby:

- MQRO_NEW_MSG_ID
- MQRO_PASS_MSG_ID
- MQRO_COPY_MSG_ID_TO_CORREL_ID
- ID_KOLEKCE_MQRO_PASS_RELACE_

Pokud nelze nezpracovatelnou zprávu zařadit do fronty, pravděpodobně proto, že je fronta nedoručených zpráv plná nebo je-li autorizace nesprávně zadána, záleží na tom, jak bude zpráva přetrvávajícím způsobem závislá. Je-li zpráva přechodná, bude zpráva vyřazena a nebude vygenerována žádná zpráva sestavy. Je-li zpráva trvalá, doručení zpráv všem odběratelům připojení, kteří naslouchají na tomto místě určení, se zastaví. Tito spotřebitelé připojení musí být zavřeni a problém vyřešen před tím, než je možné znovu vytvořit a restartovat doručení zprávy.

Je důležité definovat frontu nedoručených zpráv a pravidelně kontrolovat, zda nedošlo k žádným problémům. Zejména zajistěte, aby fronta nedoručených zpráv nedosáhla své maximální hloubky a aby maximální velikost zprávy byla dostatečně velká pro všechny zprávy.

Je-li zpráva znovu zařazena do fronty nedoručených zpráv, je jí uvedena hlavička WebSphere MQ (MQDLH). Podrobnosti o formátu rozhraní MQDLH najdete v tématu Záhlaví MQDLH-Dead-letter . Můžete identifikovat zprávy, které objekt ConnectionConsumer umístil do fronty nedoručených zpráv, nebo zprávy sestav, které generoval ConnectionConsumer , a to pomocí následujících polí:

- PutApplTyp je `MQAT_JAVA (0x1C)`
- PutApplNázev je "`MQ JMS ConnectionConsumer`"

Tato pole jsou ve frontě MQDLH zpráv ve frontě nedoručených zpráv a ve zprávách sestav MQMD. Pole zpětné vazby MQMD a pole Příčina operace MQDLH obsahují kód popisující chybu. Podrobné informace o těchto kódech najdete v tématu “Kódy příčiny a zpětné vazby v ASF” na stránce 900. Ostatní pole jsou popsána v části MQDLH-Dead-letter header.

Zpracování nezpracovatelných zpráv v ASF

V rámci služeb aplikačního serveru je zpracování nezpracovatelných zpráv ošetřeno mírně odlišně v produktu WebSphere MQ Classes for JMS.

Informace o zpracování nezpracovatelných zpráv ve třídách produktu WebSphere MQ pro platformu JMS naleznete v tématu “Zpracování nezpracovatelných zpráv v produktu IBM WebSphere MQ classes for JMS” na stránce 859.

Když používáte funkce ASF (Application Server Facilities), ConnectionConsumer, nikoli MessageConsumer, zpracuje nezpracovatelné zprávy. Funkce ConnectionConsumer znovu řadí zprávy podle vlastností QName fronty BackoutThreshold a QName BackoutRequeue.

Když aplikace používá ConnectionConsumers, okolnosti, za kterých je zpráva zálohována, závisí na relaci, kterou poskytuje aplikační server:

- Je-li relace netransakční, s parametrem AUTO_ACKNOWLEDGE nebo DUPS_OK_ACKNOWLEDGE, je zpráva vrácena pouze po chybě systému nebo v případě neočekávaného ukončení činnosti aplikace.
- Pokud relace není součástí transakce s CLIENT_ACKNOWLEDGE, nepotvrzené zprávy mohou být vráceny aplikačním serverem voláním Session.recover().

  Obvykle implementace klienta MessageListener nebo aplikační server volá funkci Message.acknowledge(). Message.acknowledge() bere na vědomí všechny zprávy doručené v relaci tak daleko.
- Je-li relace ve funkci transakce, může aplikační server volat nepotvrzené zprávy voláním funkce Session.rollback().
- Pokud aplikační server dodává aplikaci XASession, jsou zprávy potvrzovány nebo zálohovány v závislosti na distribuované transakci. Aplikační server přebírá odpovědnost za dokončení transakce.

Vestavěný poskytovatel platformy JMS v produktu WebSphere Application Server verze 5.0 a verze 5.1 zpracovává nezpracovatelné zprávy jiným způsobem, jak je popsáno pro třídy WebSphere MQ pro službu JMS. Informace o způsobu, jakým poskytovatel vloženého JMS zpracovává nezpracovatelné zprávy, naleznete v příslušné dokumentaci produktu WebSphere Application Server.

Ošetření chyb

Tento oddíl pokrývá různé aspekty ošetření chyb, včetně “Obnova z chybových stavů v ASF” na stránce 899 a “Kódy příčiny a zpětné vazby v ASF” na stránce 900.

Obnova z chybových stavů v ASF

Pokud má vlastnost ConnectionConsumer závažnou chybu, bude doručení zprávy všem uživatelům ConnectionConsumers se zájmem o stejné zastavení QLOCAL. Pokud k tomu dojde, bude upozorněn kterýkoli modul ExceptionListener , který je registrován u ovlivněného připojení. Existují dva způsoby, jak se může aplikace zotavit z těchto chybových stavů.

Typicky se vyskytne závažná chyba v této povaze, pokud ConnectionConsumer nemůže znovu zařadit zprávu do fronty nedoručených zpráv nebo se setká s chybou při čtení zpráv z QLOCAL.

Protože je upozorněn kterýkoli modul ExceptionListener , který je registrován u ovlivněného připojení, můžete je použít k identifikaci příčiny problému. V některých případech musí administrátor systému zasáhnout a vyřešit problém.

Pro zotavení z těchto chybových stavů použijte jednu z následujících metod:

- Volejte příkaz `close()` na všech ovlivněných ConnectionConsumers. Aplikace může vytvářet nové ConnectionConsumers pouze po uzavření všech ovlivněných ConnectionConsumers a všechny systémové problémy se vyřeší.
- Volejte příkaz `stop()` na všech ovlivněných připojeních. After all Connections are stopped and any system problems are resolved, the application can `start()` its Connections successfully.

Kódy příčiny a zpětné vazby v ASF

Použijte důvod a kódy zpětné vazby k určení příčiny chyby. Zde jsou uvedeny obecné kódy příčiny vygenerované položkou ConnectionConsumer .

Chcete-li určit příčinu chyby, použijte následující informace:

- Kód zpětné vazby ve všech zprávách sestav
- Kód příčiny v MQDLH všech zpráv ve frontě nedoručených zpráv

ConnectionConsumers generují následující kódy příčiny.

MQRC_BACKOUT_THRESHOLD_REACHED (0x93A; 2362)

Příčina
Zpráva dosáhla prahové hodnoty Backout definované na QLOCAL, ale není definována žádná fronta vrácení.

Na platformách, v nichž nelze definovat frontu vrácení, se zpráva dostala do prahové hodnoty vrácení definovaný systémem JMS 20.

Akce
Pokud to není požadováno, definujte frontu vrácení pro příslušnou QLOCAL. Také se podívejte na příčinu více zákuliů.

MQRC_MSG_NOT_MATCHED (0x93B; 2363)

Příčina
V systému zpráv typu point-to-point je zpráva, která neodpovídá žádnému z selektorů pro monitorování fronty ConnectionConsumers . Chcete-li zachovat výkon, je zpráva znovu zařazena do fronty nedoručených zpráv.

Akce
Chcete-li se této situaci vyhnout, ujistěte se, že ConnectionConsumers používající frontu poskytují sadu selektorů, které se zabývají všemi zprávami, nebo nastavte továrnu QueueConnectionna uchování zpráv.

Případně můžete vyšetřit zdroj zprávy.

MQRC_JMS_FORMAT_ERROR (0x93C; 2364)

Příčina
Služba JMS nemůže interpretovat zprávu ve frontě.

Akce
Prozkoumejte původ zprávy. Služba JMS obvykle doručuje zprávy neočekávaného formátu jako BytesMessage nebo TextMessage. Občas se to nezdaří, je-li zpráva velmi špatně naformátována.

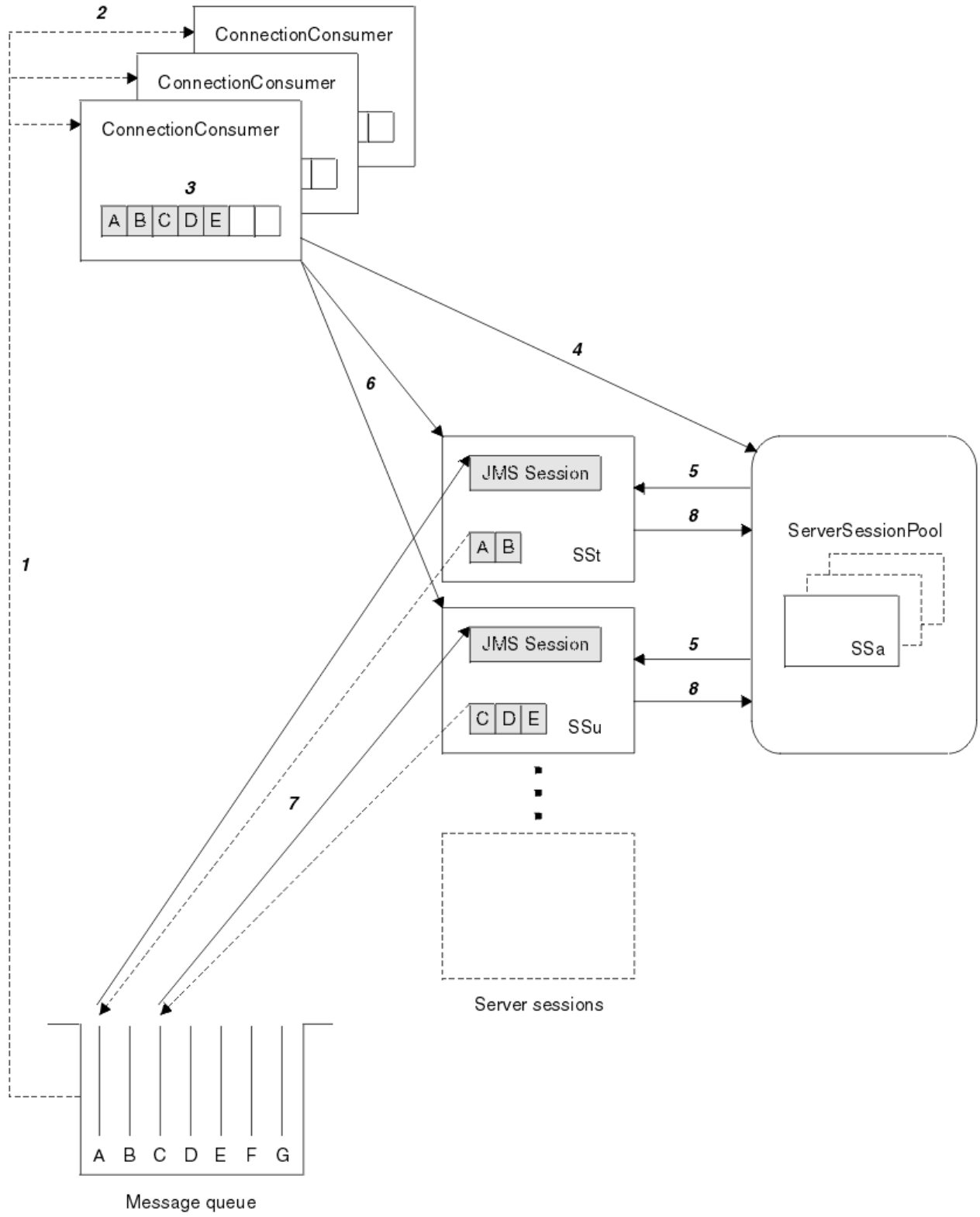
Další kódy, které se objevují v těchto polích, jsou způsobeny neúspěšným pokusem o opětné zařazení zprávy do fronty vrácení. V této situaci kód popisuje příčinu selhání funkce requeue. Chcete-li diagnostikovat příčinu těchto chyb, přečtěte si téma Kódy příčiny rozhraní API.

Pokud nelze zprávu sestavy uložit do fronty ReplyTo, vložte ji do fronty nedoručených zpráv. V této situaci je pole zpětné vazby MQMD dokončeno, jak je popsáno v tomto tématu. Pole příčiny v MQDLH vysvětluje, proč nemohla být zpráva sestavy umístěna na ReplyToQ.

Funkce fondu relací serveru v systému AFS

Toto téma shrnuje funkci fondu relací serveru.

Příkaz Obrázek 165 na stránce 901 shrnuje zásady funkcí ServerSessionPool a ServerSession .



Obrázek 165. Funkčnost ServerSessionPool a ServerSession

1. Volba ConnectionConsumers získá odkazy na zprávy z fronty.
2. Každý ConnectionConsumer vybírá specifické odkazy na zprávy.
3. Vyrovnávací paměť ConnectionConsumer obsahuje vybrané odkazy na zprávy.
4. Hodnota ConnectionConsumer vyžaduje jednu nebo více relací ServerSessions ze fondu ServerSession.

5. ServerSessions jsou alokovány ze oblasti ServerSessionPool.
6. Objekt ConnectionConsumer přiřadí odkazy na zprávy do relací ServerSessions a spustí podprocesy ServerSession spuštěné.
7. Každá ServerSession načítá odkazované zprávy z fronty. předává je metodě onMessage z objektu MessageListener , který je přidružen k relaci JMS.
8. Po dokončení zpracování se ServerSession vrátí do fondu.

Aplikační server obvykle poskytuje funkce ServerSessionPool a ServerSession .

Použití nástroje pro administraci produktu WebSphere MQ JMS

Pomocí nástroje pro administraci definujte vlastnosti osmi typů tříd produktu WebSphere MQ pro objekt JMS a uložte je do oboru názvů rozhraní JNDI. Aplikace pak mohou používat rozhraní JNDI k načtení těchto spravovaných objektů z oboru názvů.

Objekty JMS WebSphere MQ , které můžete spravovat pomocí nástroje, jsou:

- MQConnectionFactory
- Továrna MQQueueConnection
- Továrna MQTopicConnection
- MQQUEUE
- MQTopic
- MQXAConnectionFactory
- Továrna MQXAQueueConnection
- Továrna MQXATopicConnection

Podrobné informace o těchto objektech najdete v tématu “Správa objektů platformy JMS” na stránce 906 .

Typy vlastností a hodnoty, které potřebujete k použití tohoto nástroje, jsou uvedeny v části Vlastnosti objektů IBM WebSphere MQ classes for JMS.

Nástroj také umožňuje administrátorům manipulovat s podkontexty oboru názvů adresáře v rámci rozhraní JNDI. Viz “Manipulace s podkontexty s nástrojem pro administraci produktu WebSphere MQ JMS” na stránce 906.

Administrované objekty platformy JMS můžete také vytvářet a konfigurovat pomocí Průzkumníka WebSphere MQ .

Vyvolání nástroje pro administraci produktu IBM WebSphere MQ classes for JMS

Administrační nástroj má rozhraní příkazového řádku. Můžete ji použít interaktivně, nebo ji použít ke spuštění dávkového zpracování.

Interaktivní režim poskytuje příkazový řádek, kde můžete zadat příkazy administrace. V dávkovém režimu obsahuje příkaz ke spuštění nástroje název souboru, který obsahuje příkazový skript administrace.

Interaktivní režim

Chcete-li spustit nástroj v interaktivním režimu, zadejte následující příkaz:

```
JMSAdmin [-t] [-v] [-cfg config_filename]
```

kde:

-t
Povolí trasování (výchozí je trasování vypnuto).

Trasovací soubor je generován v systémech "%MQ_JAVA_DATA_PATH%"\errors (Windows) nebo /var/mqm/trace (UNIX). Název trasovacího souboru je ve tvaru:

mqjms_*PID*.trc

kde *PID* je ID procesu prostředí JVM.

-v
Produkuje výstup s komentářem (výchozí hodnota je terse output)

-cfg název_konfiguračního_souboru
Název alternativního konfiguračního souboru. Je-li tento parametr vynechán, použije se výchozí konfigurační soubor JMSAdmin.config. (Viz “Konfigurace nástroje pro administraci služby JMS” na stránce 903)

Zobrazí se příkazový řádek, který označuje, že je nástroj připraven přijímat příkazy administrace. Tato výzva se nejprve zobrazí jako:

```
InitCtx>
```

indikující, že aktuální kontext (tj. kontext rozhraní JNDI, na který se aktuálně odkazují všechny operace pojmenování a adresářů) je počáteční kontext definovaný v konfiguračním parametru PROVIDER_URL (viz “Konfigurace nástroje pro administraci služby JMS” na stránce 903).

Při procházení oboru názvů adresáře se výzva k zobrazení této výzvy projeví, takže výzva k zadání vždy zobrazí aktuální kontext.

Dávkový režim

Chcete-li spustit nástroj v dávkovém režimu, zadejte příkaz:

```
JMSAdmin <test.scp
```

kde *test.scp* je skriptový soubor, který obsahuje příkazy administrace (viz “Příkazy administrace v nástroji pro administraci produktu WebSphere MQ JMS” na stránce 905). Poslední příkaz v souboru musí být příkaz END .

Konfigurace nástroje pro administraci služby JMS

Konfigurační soubor produktu WebSphere MQ JMS používá konfigurační soubor k nastavení hodnot určitých vlastností. K dispozici je ukázkový soubor, který můžete přizpůsobit svému systému.

Konfigurační soubor je prostý textový soubor, který se skládá ze sady dvojic klíč-hodnota oddělených znakem rovnítka (=). To je zobrazeno v následujícím příkladu:

```
#Set the service provider
    INITIAL_CONTEXT_FACTORY=com.sun.jndi.ldap.LdapCtxFactory
#Set the initial context
    PROVIDER_URL=ldap://polaris/o=ibm_us,c=us
#Set the authentication type
    SECURITY_AUTHENTICATION=none
```

(A # v prvním sloupci řádku označuje komentář, nebo řádek, který se nepoužívá.)

Ukázkový konfigurační soubor je dodáván s produktem WebSphere MQ. Tento soubor se nazývá JMSAdmin.configa nachází se v adresáři <MQ_JAVA_INSTALL_PATH>/bin . Upravte tento soubor tak, aby odpovídal nastavení vašeho systému.

Nakonfigurujte nástroj pro administraci s hodnotami pro následující vlastnosti:

INITIAL_CONTEXT_FACTORY
Poskytovatel služby, kterého nástroj používá. Podporované hodnoty této vlastnosti jsou následující:

- com.sun.jndi.ldap.LdapCtxFactory (pro LDAP)

- com.sun.jndi.fscontext.RefFSContextFactory (pro kontext systému souborů)

Můžete také použít továrnu InitialContextFactory, která není uvedena v předchozím seznamu. Další podrobnosti naleznete v části "Použití neuvedeného továrny InitialContextFactory s nástrojem pro administraci produktu WebSphere MQ JMS" na stránce 904.

PROVIDER_URL
Adresa URL počátečního kontextu relace; kořen všech operací JNDI provedených nástrojem. Jsou podporovány dvě formy této vlastnosti:

- ldap://název_hostitele/kontextový_název
- file: [jednotka:] /název_cesty

Formát adresy URL protokolu LDAP se může lišit v závislosti na poskytovateli LDAP. Další informace naleznete v dokumentaci k protokolu LDAP.

ZABEZPEČENÍ_ZABEZPEČENÍ
Určuje, zda rozhraní JNDI předá pověření zabezpečení poskytovateli služeb. Tato vlastnost se používá pouze tehdy, je-li použit poskytovatel služby LDAP. Tato vlastnost může mít jednu ze tří hodnot:

- žádné (anonymní ověření)
- jednoduché (jednoduché ověření)
- CRAM-MD5 (mechanizmus ověřováníCRAM-MD5)

Není-li zadána platná hodnota, vlastnost se standardně nastaví na hodnotu none. Další informace o zabezpečení pomocí nástroje pro administraci viz "Konfigurace zabezpečení pro nástroj pro administraci služby JMS" na stránce 904 .

Tyto vlastnosti jsou nastaveny v konfiguračním souboru. Když vyvoláte nástroj, můžete tuto konfiguraci zadat pomocí parametru příkazového řádku `-cfg` , jak je popsáno v části "Vyvolání nástroje pro administraci produktu IBM WebSphere MQ classes for JMS" na stránce 902. Pokud nezadáte žádný název konfiguračního souboru, nástroj se pokusí načíst výchozí konfigurační soubor (`JMSAdmin.config`). Hledá tento soubor nejprve v aktuálním adresáři a potom v adresáři `<MQ_JAVA_INSTALL_PATH>/bin` , kde `<MQ_JAVA_INSTALL_PATH>` je cesta k vašim třídám WebSphere MQ pro instalaci JMS.

Použití neuvedeného továrny InitialContextFactory s nástrojem pro administraci produktu WebSphere MQ JMS

Jsou podporovány dva hodnoty továrny InitialContextFactory. Další kontexty JNDI můžete použít nastavením parametrů v konfiguračním souboru administrace platformy JMS.

Administrační nástroj lze použít k připojení k jiným kontextům JNDI než těm, které jsou uvedeny v produktu "Konfigurace nástroje pro administraci služby JMS" na stránce 903 , pomocí tří parametrů definovaných v konfiguračním souboru JMSAdmin.

Chcete-li použít jinou továrnu InitialContext, postupujte takto:

1. Nastavte vlastnost INITIAL_CONTEXT_FACTORY na požadovaný název třídy.
2. Definujte chování továrny InitialContextFactory pomocí vlastností USE_INITIAL_DIR_CONTEXT, NAME_PREFIX a NAME_READABILITY_MARKER.

Nastavení pro tyto vlastnosti jsou popsány v komentářích ukázkového konfiguračního souboru.

Pokud použijete některou z podporovaných hodnot INITIAL_CONTEXT_FACTORY, není třeba zde uvedené tři vlastnosti definovat. Avšak můžete jim dát hodnoty, aby potlačují výchozí nastavení systému. Vynecháte-li jeden nebo více z těchto tří vlastností továrny InitialContext, nástroj pro administraci poskytne vhodné výchozí hodnoty založené na hodnotách ostatních vlastností.

Konfigurace zabezpečení pro nástroj pro administraci služby JMS

Vlastnost SECURITY_AUTHENTICATION použijte k určení, zda jsou pověření zabezpečení předána poskytovateli služeb.

Vlastnost SECURITY_AUTHENTICATION je popsána v tématu "Konfigurace nástroje pro administraci služby JMS" na stránce 903. Jeho účinek je následující:

- Nastavíte-li tento parametr na hodnotu none, rozhraní JNDI nepředá žádné pověření zabezpečení poskytovateli služby a provede se *anonymní ověření* .
- Nastavíte-li tento parametr na hodnotu `simple` nebo `CRAM-MD5`, budou pověření zabezpečení předávána prostřednictvím rozhraní JNDI základního poskytovatele služeb. Tato bezpečnostní pověření jsou ve formě rozlišovacího jména uživatele (DN uživatele) a hesla.

Jsou-li požadována pověření zabezpečení, budete při inicializaci nástroje vyzváni k jejich zadání. Vyhněte se tomu nastavením vlastností PROVIDER_USERDN a PROVIDER_PASSWORD v konfiguračním souboru JMSAdmin.

Poznámka: Pokud tyto vlastnosti nepoužijete, vypíše se na obrazovku text zadaný *včetně hesla*. To může mít dopad na zabezpečení.

Nástroj neprovádí vlastní ověření; úloha je delegována na server LDAP. Administrátor serveru LDAP musí nastavit a udržovat přístupová oprávnění k různým částem adresáře. Další informace naleznete v dokumentaci k protokolu LDAP. Pokud ověření selže, nástroj zobrazí odpovídající chybovou zprávu a ukončí se.

Podrobnější informace o zabezpečení a rozhraní JNDI najdete v dokumentaci na webových stránkách Sun's Java (https://java.sun.com).

Příkazy administrace v nástroji pro administraci produktu WebSphere MQ JMS

Nástroj pro správu přijímá příkazy skládající se z příkazového slova a jeho příslušných parametrů.

Když se zobrazí příkazový řádek, nástroj je připraven přijmout příkazy. Příkazy administrace jsou obvykle v následujícím tvaru:

```
verb [param]*
```

kde **verb** je jedním z administračních příkazových slov uvedených v Tabulka 133 na stránce 905. Všechny platné příkazy obsahují jedno příkazové slovo, které se objevuje na začátku příkazu buď ve standardním, nebo v krátkém tvaru.

Parametry, které může příkaz provést, závisí na slově. Příkazové slovo END například nemůže přijímat žádné parametry, ale slovo DEFINE může mít libovolný počet parametrů. Podrobnosti o příkazových slovech, které mají obsahovat alespoň jeden parametr, jsou popsány v souvisejících tématech.

Tabulka 133. Příkazová slova

Sloveso	**Krátký formát**	**Popis**
ALTER	KLÁVESA ALT	Změnit alespoň jednu z vlastností spravovaného objektu
Definice	DEF	Vytvoření a uložení spravovaného objektu nebo vytvoření dílčího kontextu
DISPLAY	DIS	Zobrazit vlastnosti jednoho nebo více uložených spravovaných objektů nebo obsah aktuálního kontextu
ODSTRANIT	DEL	Odebrat jeden nebo více spravovaných objektů z oboru názvů nebo odebrat prázdný podkontext
CHANGE	chg	Pozměnit aktuální kontext tak, aby uživatel mohl procházet obor názvů adresáře kdekoli pod počátečním kontextem (nevyřízená bezpečnostní prověrka)
COPY	CP	Vytvořit kopii uloženého spravovaného objektu a uložit jej pod alternativní název
MOVE	MV	Změnit název, pod kterým je spravovaný objekt uložen.
END		Zavřete nástroj pro administraci.

Názvy příkazů Verb nejsou citlivé na velikost písmen.

Chcete-li příkazy ukončit, je třeba stisknout klávesu CR. Tuto operaci však můžete potlačit zadáním znaku plus (+) přímo před návratem vozíku. To vám umožní zadat víceřádkové příkazy, jak je zobrazeno v následujícím příkladu:

```
DEFINE Q(BookingsInputQueue) +
       QMGR(QM.POLARIS.TEST) +
       QUEUE(BOOKINGS.INPUT.QUEUE) +
       PORT(1415) +
       CCSID(437)
```

Řádky začínající libovolným z následujících znaků jsou považovány za komentáře a jsou ignorovány: * #/.

Manipulace s podkontexty s nástrojem pro administraci produktu WebSphere MQ JMS

Pomocí příkazových slov **CHANGE**, **DEFINE**, **DISPLAY** a **DELETE** můžete manipulovat s podkontexty oboru názvů adresáře.

Použití těchto příkazových slov je popsáno v části Tabulka 134 na stránce 906.

Tabulka 134. Syntaxe a popis příkazů používaných k manipulaci s podkontexty

Syntaxe příkazu	**Popis**
DEFINE CTX (ctxName)	Pokusí se o vytvoření podřízeného dílčího kontextu aktuálního kontextu s názvem ctxName. Pokud již existuje narušení zabezpečení, dojde k selhání zabezpečení, pokud již existuje, nebo není-li zadaný název platný.
ZOBRAZIT CTX	Zobrazí obsah aktuálního kontextu. Spravované objekty jsou anotovány pomocí a, dílčích kontextů s `[D]`. Zobrazí se také typ Java každého objektu.
ODSTRANIT CTX (ctxName)	Pokusy o odstranění podřízeného kontextu aktuálního kontextu s názvem ctxName. Pokud kontext nebyl nalezen, je neprázdný kontext nebo došlo k narušení zabezpečení.
CHANGE CTX (ctxName)	Pozměňuje aktuální kontext, takže se nyní odkazuje na podřízený kontext s názvem ctxName. Je možné zadat jednu ze dvou speciálních hodnot ctxName : **= NAHORU** přesunout na nadřízený prvek aktuálního kontextu **= INICIALIZACE** přesun přímo do počátečního kontextu Selhání, pokud určený kontext neexistuje, nebo pokud došlo k narušení zabezpečení.

Správa objektů platformy JMS

Tento oddíl popisuje osm typů objektů, které může administrační nástroj zpracovat. Obsahuje podrobnosti o každé z jejich konfigurovatelných vlastností a příkazových slov, které s nimi mohou manipulovat.

Administrované objekty platformy JMS můžete také vytvářet a konfigurovat pomocí Průzkumníka WebSphere MQ .

Typy objektů JMS

V tabulce jsou zobrazeny osm typů spravovaných objektů.

Sloupec Klíčové slovo zobrazuje řetězce, které můžete nahradit pro *TYPE* v příkazech zobrazených v Tabulka 136 na stránce 908.

Tabulka 135. Typy objektů platformy JMS, které jsou obsluhovány nástrojem pro administraci

Typ objektu	**Klíčové slovo**	**Popis**
MQConnectionFactory	CF	Implementace rozhraní JMS ConnectionFactory produktu WebSphere MQ . Představuje objekt továrny pro vytvoření připojení v doménách typu point-to-point i publikování/odběr.
Továrna MQQueueConnection	QCF	The WebSphere MQ implementation of the JMS QueueConnectionFactory interface. To představuje objekt továrny pro vytvoření připojení v doméně dvoubodového spojení.
Továrna MQTopicConnection	TCF	The WebSphere MQ implementation of the JMS TopicConnectionFactory interface. Tento objekt představuje objekt továrny pro vytváření připojení v doméně publikování/odběru.
MQQUEUE	Q	Implementace rozhraní JMS (WebSphere MQ). Představuje místo určení pro zprávy v rámci domény typu point-to-point.
MQTopic	T	Implementace rozhraní tématu JMS produktu WebSphere MQ . Představuje místo určení pro zprávy v doméně publikování/odběru.
MQXAConnectionFactory"1" na stránce 907	XACF	Implementace rozhraní JMS XAConnectionFactory produktu WebSphere MQ . Představuje objekt továrny pro vytvoření připojení v doménách typu point-to-point i publikování/odběr a kde připojení používají verze XA pro třídy JMS.
Továrna MQXAQueueConnectionFactory"1" na stránce 907	XAQCF	The WebSphere MQ implementation of the JMS XAQueueConnectionFactory interface. To představuje objekt továrny pro vytvoření připojení v doméně dvoubodového spojení, které používají verze XA tříd JMS.
Továrna MQXATopicConnectionFactory"1" na stránce 907	XATCF	The WebSphere MQ implementation of the JMS XATopicConnectionFactory interface. Představuje objekt továrny pro vytváření připojení v doméně publikování/odběru, které používají verze XA pro třídy JMS.

Poznámka:

1. Tyto třídy jsou k dispozici pro použití od dodavatelů aplikačních serverů. Je nepravděpodobné, že by byly pro programátory aplikací přímo užitečné.

Slovesa používané s objekty platformy JMS

Chcete-li manipulovat se spravovanými objekty v oboru názvů adresáře, můžete použít příkazy `ALTER`, `DEFINE`, `DISPLAY`, `DELETE`, `COPY`a `MOVE` .

Tabulka 136 na stránce 908 shrnuje použití těchto příkazových slov. Nahraďte řetězec *TYPE* klíčovým slovem, které představuje požadovaný spravovaný objekt, jak je uvedeno v tématu Tabulka 135 na stránce 907.

Tabulka 136. Syntaxe a popis příkazů používaných k manipulaci se spravovanými objekty

Syntaxe příkazu	**Popis**
ALTER *TYPE*(název) [vlastnost] *	Pokusí se aktualizovat vlastnosti administrovaného objektu s dodanými objekty. Pokud došlo k narušení zabezpečení, selhání v případě, že uvedený objekt nebyl nalezen, nebo nejsou-li zadané nové vlastnosti platné.
DEFINE *TYP*(název) [vlastnost] *	Pokouší se vytvořit spravovaný objekt typu *TYPE* s dodanými vlastnostmi a uložit jej pod názvem name v aktuálním kontextu. Pokud je poskytnutý název neplatný nebo objekt s daným názvem existuje, nebo pokud zadané vlastnosti nejsou platné, došlo k selhání zabezpečení.
DISPLAY *TYP*(název)	Zobrazí vlastnosti spravovaného objektu typu *TYPE*s vazbou pod názvem name v aktuálním kontextu. Selhání, pokud objekt neexistuje, nebo pokud došlo k narušení zabezpečení.
DELETE *TYPE*(název)	Pokusí se odebrat administrovaný objekt typu *TYPE*, který má název name, z aktuálního kontextu. Selhání, pokud objekt neexistuje, nebo pokud došlo k narušení zabezpečení.
COPY *TYP*(nameA) *TYP*(nameB)	Vytvoří kopii spravovaného objektu typu *TYPE*s názvem nameAa pojmenovává se kopie nameB. Toto vše se vyskytuje v rozsahu aktuálního kontextu. Pokud objekt, který má být kopírován, neexistuje, existuje-li objekt s názvem nameB , nebo pokud dojde k narušení zabezpečení.
MOVE *TYPE*(nameA) *TYP*(nameB)	Přesouvá (přejmenovává) spravovaný objekt typu *TYPE*s názvem nameAna nameB. Toto vše se vyskytuje v rozsahu aktuálního kontextu. Pokud objekt, který má být přesunut, neexistuje, existuje-li objekt s názvem nameB , nebo pokud došlo k narušení zabezpečení.

Vytváření objektů s použitím nástroje pro administraci produktu WebSphere MQ JMS

Vytvořit objekty a uložit je do oboru názvů JNDI pomocí příkazu DEFINE,

Použijte následující syntaxi příkazu:

```
DEFINE TYPE(name) [property]*
```

To znamená, že příkazové slovo DEFINE je následováno odkazem na spravovaný objekt *TYPE*(name) , následované nulou nebo více *vlastnostmi* (viz Vlastnosti objektů IBM WebSphere MQ classes for JMS).

Aspekty pojmenování LDAP pro objekty platformy JMS

Chcete-li ukládat své objekty v prostředí LDAP, musíte jim dát názvy, které odpovídají určitým konvencím. Administrativní nástroj vám může pomoci dodržovat konvence pojmenování tak, že přidáte výchozí předponu.

Jedna konvence pojmenování je taková, že názvy objektu a podkontextu musí obsahovat předponu, jako například cn= (obecný název) nebo ou= (organizační jednotka).

Administrační nástroj zjednodušuje použití poskytovatelů služeb LDAP tím, že vám umožňuje odkazovat na názvy objektů a kontextů bez předpony. Pokud nezadáte předponu, nástroj automaticky přidá výchozí předponu k názvu, který dodáte. Pro protokol LDAP je to cn=.

Výchozí předponu můžete změnit nastavením vlastnosti NAME_PREFIX v konfiguračním souboru JMSAdmin, jak je popsáno v tématu “Použití neuvedeného továrny InitialContextFactory s nástrojem pro administraci produktu WebSphere MQ JMS” na stránce 904.

To je zobrazeno v následujícím příkladu.

```
InitCtx> DEFINE Q(testQueue)

InitCtx> DISPLAY CTX

    Contents of InitCtx

      a  cn=testQueue              com.ibm.mq.jms.MQQueue

     1 Object(s)
       0 Context(s)
       1 Binding(s), 1 Administered
```

Přestože zadaný název objektu (`testQueue`) nemá předponu, nástroj automaticky přidá jeden k zajištění shody s konvencemi pojmenování LDAP. Podobně, odeslání příkazu `DISPLAY Q(testQueue)` také způsobí, že se přidá tato předpona.

Možná budete muset nakonfigurovat server LDAP pro ukládání objektů Java. Informace, které vám pomohou při konfiguraci této konfigurace, najdete v dokumentaci k serveru LDAP.

Ukázkové chybové stavy vytvářející objekt platformy JMS

Při vytváření objektu může vzniknout řada běžných chybových stavů.

Níže jsou uvedeny příklady těchto chybových stavů:

CipherSpec mapována na CipherSuite

```
InitCtx/cn=Trash> DEFINE QCF(testQCF) SSLCIPHERSUITE(RC4_MD5_US)
  WARNING: Converting CipherSpec RC4_MD5_US to
  CipherSuite SSL_RSA_WITH_RC4_128_MD5
```

Neplatná vlastnost pro objekt

```
InitCtx/cn=Trash> DEFINE QCF(testQCF) PRIORITY(4)
  Unable to create a valid object, please check the parameters supplied
  Invalid property for a QCF: PRI
```

Neplatný typ hodnoty vlastnosti

```
InitCtx/cn=Trash> DEFINE QCF(testQCF) CCSID(english)
  Unable to create a valid object, please check the parameters supplied
  Invalid value for CCS property: English
```

Konflikt vlastností-klient/bin.

```
InitCtx/cn=Trash> DEFINE QCF(testQCF) HOSTNAME(polaris.hursley.ibm.com)
  Unable to create a valid object, please check the parameters supplied
  Invalid property in this context: Client-bindings attribute clash
```

Konflikt vlastností-inicializace uživatelské procedury

```
InitCtx/cn=Trash> DEFINE QCF(testQCF) SECEXITINIT(initStr)
  Unable to create a valid object, please check the parameters supplied
  Invalid property in this context: ExitInit string supplied
  without Exit string
```

Hodnota vlastnosti mimo platný rozsah

```
InitCtx/cn=Trash> DEFINE Q(testQ) PRIORITY(12)
  Unable to create a valid object, please check the parameters supplied
  Invalid value for PRI property: 12
```

Neznámá vlastnost

```
InitCtx/cn=Trash> DEFINE QCF(testQCF) PIZZA(ham and mushroom)
  Unable to create a valid object, please check the parameters supplied
  Unknown property: PIZZA
```

Následují příklady chybových stavů, které se mohou objevit na systému Windows při hledání administrovaných objektů JNDI z aplikace JMS.

1. Používáte-li poskytovatele JNDI WebSphere , com.ibm.websphere.naming.WsnInitialContextFactory, musíte použít dopředné lomítko (/) pro přístup ke spravovaným objektům definovaným v dílčích kontextech; například jms/MyQueueNázev. Použijete-li zpětné lomítko (\), dojde k výjimce InvalidName.
2. Pokud používáte poskytovatele JNDI Sun JNDI, com.sun.jndi.fscontext.RefFSContextFactory, musíte použít zpětné lomítko (\) pro přístup ke spravovaným objektům definovaným v dílčích kontextech; například ctx1\\fred. Použijete-li dopředné lomítko (/), dojde k NameNotFoundException .

Použití Průzkumníka produktu WebSphere MQ pro konfiguraci JMS

Použijte grafické uživatelské rozhraní produktu IBM WebSphere MQ Explorer k vytvoření objektů JMS z objektů WebSphere MQ a objektů WebSphere MQ z objektů platformy JMS a také pro administraci a monitorování dalších objektů produktu WebSphere MQ .

Než začnete

Před vytvořením a konfigurací spravovaných objektů platformy JMS s produktem WebSphere MQ Explorer přidejte počáteční kontext a definujte kořenový adresář oboru názvů JNDI, ve kterém jsou objekty platformy JMS uloženy ve službě pro správu pojmenování a adresářů. Další informace naleznete v uživatelské podpoře produktu IBM WebSphere MQ Explorer pro spravované objekty platformy JMS.

Informace o této úloze

S Průzkumníkem IBM WebSphere MQ můžete provádět následující úlohy buď kontextově z existujícího objektu v Průzkumníku IBM WebSphere MQ , nebo z průvodce vytvořením nového objektu. Příklady typických úloh produktu WebSphere MQ Explorer pro některé typické úlohy najdete v nápovědě k programu Průzkumník produktu WebSphere MQ .

Procedura

- Vytvořte továrnu na připojení rozhraní JMS z libovolného z následujících objektů produktu WebSphere MQ :
  a) Správce front produktu WebSphere MQ , ať už na lokálním počítači nebo na vzdáleném systému.
  b) Kanál produktu WebSphere MQ
  c) Listener WebSphere MQ
- Přidejte správce front WebSphere MQ do produktu WebSphere MQ Explorer pomocí továrny připojení platformy JMS.
- Vytvoření fronty platformy JMS z fronty produktu WebSphere MQ
- Vytvoření fronty produktu WebSphere MQ z fronty JMS
- Vytvořte téma JMS z tématu WebSphere MQ , které může být objektem produktu WebSphere MQ nebo dynamickým tématem.
- Vytvoření tématu WebSphere MQ z tématu JMS

Použití balíku záhlaví produktu WebSphere MQ

Balík záhlaví produktu WebSphere MQ poskytuje sadu nápovědných rozhraní a tříd, které můžete použít k manipulaci se záhlavími WebSphere MQ zprávy. Obvykle se používá balík záhlaví produktu WebSphere

MQ , protože chcete provádět administrativní služby pomocí příkazového serveru (pomocí zpráv PCF (Programmable Command Format)).

Informace o této úloze

Balík produktu WebSphere MQ Headers je umístěn v balících `com.ibm.mq.headers` a `com.ibm.mq.pcf` . Tuto funkci můžete použít pro obě dvě alternativní rozhraní API, která produkt WebSphere MQ poskytuje pro použití v aplikacích Java:

- WebSphere MQ classes for Java (also referred to as WebSphere MQ Headers Base Java).
- Třídy WebSphere MQ pro službu JMS (WebSphere MQ pro službu JMS) jsou označovány také jako WebSphere MQ JMS).

Aplikace WebSphere MQ Base Java obvykle manipulují s objekty MQMessage a třídy podpory záhlaví mohou s těmito objekty přímo komunikovat, protože nativně chápou základní rozhraní jazyka Java produktu WebSphere MQ .

V produktu WebSphere MQ JMS je informačním obsahem zprávy zpravidla řetězec nebo objekt bajtového pole, kterému lze manipulovat s proudy DataInput a DataOutput . Balík záhlaví produktu WebSphere MQ lze použít pro interakci s těmito datovými proudy a je vhodný pro manipulaci se zprávami produktu MQ , které jsou odesílány a přijímány aplikacemi WebSphere MQ JMS.

Proto ačkoli balík záhlaví produktu WebSphere MQ obsahuje odkazy na balík produktu WebSphere MQ Base Java, je určen také pro použití v rámci aplikací WebSphere MQ JMS a je vhodný pro použití v prostředí Java Platform, Enterprise Edition (Java EE).

Typickým způsobem, jak můžete použít balík WebSphere MQ Headers, je práce s administračními zprávami ve formátu PCF (Programmable Command Format), například z následujících důvodů:

- Přístup k podrobnostem o prostředku produktu WebSphere MQ .
- Chcete-li monitorovat hloubku fronty.
- Zablokování přístupu k frontě.

Při použití zpráv PCF s rozhraním API služby JMS produktu WebSphere MQ lze tento způsob administrace prostředků zaměřených na aplikaci provádět z aplikací produktu Java EE , aniž by bylo nutné použít základní rozhraní Java API produktu WebSphere MQ .

Procedura

- Chcete-li použít balík záhlaví produktu WebSphere MQ k manipulaci se záhlavími zpráv pro třídy WebSphere MQ pro prostředí Java, prohlédněte si téma “Použití s třídami produktu WebSphere MQ pro prostředí Java” na stránce 911.
- Informace o použití balíku záhlaví produktu WebSphere MQ k manipulaci se záhlavími zpráv pro rozhraní JMS naleznete v tématu “Použití s třídami produktu WebSphere MQ pro službu JMS” na stránce 912.

Použití s třídami produktu WebSphere MQ pro prostředí Java

Třídy WebSphere MQ pro aplikace v jazyce Java obvykle manipulují s objekty MQMessage a třídy podpory záhlaví mohou s těmito objekty přímo komunikovat, protože nativně chápou třídy produktu WebSphere MQ pro rozhraní Java.

Informace o této úloze

Produkt WebSphere MQ nabízí některé ukázkové aplikace, které demonstrují použití balíku WebSphere MQ Headers spolu s produktem WebSphere MQ Base Java API (WebSphere MQ classes for Java).

Ukázky zobrazují dvě věci:

- Jak vytvořit zprávu PCF pro provedení administrativní akce a analýzu zprávy odezvy.
- Jak odeslat tuto zprávu PCF pomocí tříd produktu WebSphere MQ pro prostředí Java.

V závislosti na použité platformě jsou tyto ukázky instalovány v adresáři `pcf` v adresáři `samples` nebo `tools` v instalaci produktu WebSphere MQ (viz “Instalační adresáře pro třídy produktu WebSphere MQ pro prostředí Java” na stránce 635).

Postup

1. Vytvořte zprávu PCF, abyste provedli administrativní akci a analyzoval zprávu odpovědi.
2. Odešlete tuto zprávu PCF pomocí tříd produktu WebSphere MQ pro prostředí Java.

Související pojmy

“Zpracování záhlaví zpráv produktu WebSphere MQ s třídami produktu WebSphere MQ pro prostředí Java” na stránce 656
Poskytují se třídy Java představující různé typy záhlaví zprávy. Poskytují se také dvě pomocné třídy.

“Zpracování zpráv PCF s třídami produktu WebSphere MQ pro prostředí Java” na stránce 661
Třídy Java jsou poskytovány pro vytváření a analýzu zpráv strukturovaných PCF a pro usnadnění odesílání požadavků PCF a shromažďování odpovědí PCF.

Použití s třídami produktu WebSphere MQ pro službu JMS

Chcete-li používat záhlaví produktu WebSphere MQ spolu s třídami produktu WebSphere MQ pro platformu JMS, budete provádět stejné základní kroky jako pro třídy WebSphere MQ pro prostředí Java. Zprávu PCF lze vytvořit a odezvu analyzovat přesně stejným způsobem pomocí balíku záhlaví produktu WebSphere MQ a stejného vzorového kódu jako pro třídy WebSphere MQ pro prostředí Java.

Informace o této úloze

Chcete-li odeslat zprávu PCF pomocí rozhraní API produktu WebSphere MQ , musí být informační obsah zprávy zapsán do zprávy v bajtech platformy JMS a odeslán pomocí standardních rozhraní JMS API. Jediná úvaha je, že zpráva nesmí obsahovat záhlaví JMS RFH2 ani žádná jiná záhlaví se specifickými hodnotami v MQMD.

Chcete-li odeslat zprávu PCF, proveďte následující kroky. Způsob vytvoření zprávy PCF a informace extrahované ze zprávy odpovědi jsou stejné jako pro třídy produktu WebSphere MQ pro jazyk Java (viz “Použití s třídami produktu WebSphere MQ pro prostředí Java” na stránce 911).

Postup

1. Vytvořte místo určení fronty JMS, které reprezentuje SYSTEM.ADMIN.COMMAND.QUEUE.

   Aplikace JMS produktu WebSphere MQ odesílají zprávy PCF na server SYSTEM.ADMIN.COMMAND.QUEUEa potřebují mít 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 server WebSphere Application Server, je třeba tyto vlastnosti definovat jako přizpůsobené vlastnosti v rámci cíle.

   Chcete-li vytvořit místo určení programově z určité aplikace, použijte následující kód:

   ```
   Queue q1 = session.createQueue("SYSTEM.ADMIN.COMMAND.QUEUE");
   ((MQQueue) q1).setIntProperty(WMQConstants.WMQ_MESSAGE_BODY,
       WMQConstants.WMQ_MESSAGE_BODY_MQ);
   ((MQQueue) q1).setMQMDWriteEnabled(true);
   ```

2. Převeďte zprávu PCF na zprávu v bajtech JMS obsahující správné hodnoty MQMD.

   Je třeba vytvořit zprávu v bajtech JMS a do ní je zapsána zpráva PCF. Je třeba vytvořit frontu odpovědí, ale tato potřeba nemá žádná specifická nastavení.

   Následující úsek vzorového kódu ukazuje, jak vytvořit zprávu o bajtech platformy JMS a zapsat objekt com.ibm.mq.headers, objektpcf.PCFMessage do něj. Objekt PCFMessage (pcfCmd) byl dříve sestaven

pomocí balíku záhlaví produktu WebSphere 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 JMS API.
4. Převeďte zprávu odpovědi na zprávu PCF na zpracování.

   Chcete-li načíst zprávu odpovědi a zpracovat ji jako zprávu PCF, použijte následující kód:

```
 // Get the message back
BytesMessage msg = (BytesMessage) consumer.receive();

// get the size of the bytes message & read into an array
int bodySize = (int) msg.getBodyLength();
byte[] data = new byte[bodySize];
msg.readBytes(data);

// Read into Stream and DataInput Stream
ByteArrayInputStream bais = new ByteArrayInputStream(data);
DataInput dataInput = new DataInputStream(bais);

// Pass to PCF Message to process
PCFMessage response = new PCFMessage(dataInput);
```

Související pojmy

“Zprávy JMS” na stránce 780

Zprávy JMS se skládají ze záhlaví, vlastností a těla. Platforma JMS definuje pět typů těla zprávy.

Použití webových služeb v produktu WebSphere MQ

Můžete vyvíjet aplikace produktu IBM WebSphere MQ pro webové služby pomocí přenosu IBM WebSphere MQ pro protokol SOAP nebo mostu produktu IBM WebSphere MQ pro protokol HTTP.

Přenos produktu IBM WebSphere MQ pro protokol SOAP poskytuje transport JMS pro protokol SOAP. Přenos produktu IBM WebSphere MQ pro SOAP je také integrován do jiných prostředí, jako je například Microsoft Windows Communication Foundation, WebSphere Application Server a CICS Transaction Server.

Další informace o přenosu protokolu SOAP v produktu IBM WebSphere MQ naleznete v tématu “Přenos WebSphere MQ pro SOAP” na stránce 914.

Při použití mostu IBM WebSphere MQ pro protokol HTTP mohou aplikace klienta vyměňovat zprávy s produktem IBM WebSphere MQ bez nutnosti instalovat klienta WebSphere MQ MQI. Produkt WebSphere MQ můžete volat z libovolné platformy nebo jazyka s možností HTTP.

Další informace o mostu produktu IBM WebSphere MQ pro protokol HTTP naleznete v tématu “Most WebSphere MQ pro protokol HTTP” na stránce 990.

Související pojmy

“Koncepty vývoje aplikací” na stránce 7
K zápisu aplikací IBM WebSphere MQ můžete použít volbu procedurálních nebo objektově orientovaných jazyků. Informace o konceptech produktu IBM WebSphere MQ , které jsou užitečné pro vývojáře aplikací, naleznete v odkazech v tomto tématu.

“Rozhodování o tom, jaký programovací jazyk použít” na stránce 75
Tyto informace použijte k vyhledání informací o programovacích jazycích a rámcích podporovaných produktem IBM WebSphere MQa o některých aspektech jejich použití.

“Návrh aplikací produktu IBM WebSphere MQ” na stránce 86
Když se rozhodnete, jak aplikace mohou využívat výhody platforem a prostředí, které máte k dispozici, musíte se rozhodnout, jak budete používat funkce nabízené produktem WebSphere MQ.

“Ukázka programů WebSphere MQ” na stránce 92
Tato kolekce témat slouží k získání informací o ukázkových programech WebSphere MQ na různých platformách.

“Psaní front aplikace” na stránce 187
Prostřednictvím těchto informací můžete získat informace o vytváření aplikací pro řazení do front, připojování a odpojování od správce front, publikování a odběru a otevírání a zavírání objektů.

“Tvorba klientských aplikací” na stránce 337
Co potřebujete vědět, chcete-li zapisovat klientské aplikace do produktu WebSphere MQ.

“Zapisování aplikací typu publikování/odběr” na stránce 266
Začněte zapisovat do aplikací WebSphere MQ publish/subscribe.

“Sestavení aplikace IBM WebSphere MQ” na stránce 412
Tyto informace použijte k seznámení se s aplikací produktu IBM WebSphere MQ na různých platformách.

“Obsluha chyb programu” na stránce 529
Tyto informace vysvětlují chyby související s voláními MQI MQI buď při volání, nebo při doručení její zprávy do konečného cíle.

Přenos WebSphere MQ pro SOAP

Přenos protokolu WebSphere MQ pro protokol SOAP poskytuje transport JMS pro protokol SOAP. Přenos protokolu WebSphere MQ pro SOAP je také integrován do jiných prostředí, jako je například Microsoft Windows Communication Foundation, WebSphere Application Server a CICS Transaction Server.

Introduction to IBM WebSphere MQ transport for SOAP

Přenos produktu IBM WebSphere MQ pro protokol SOAP poskytuje transport JMS pro protokol SOAP. Odesilatel a modul listener protokolu SOAP produktu WebSphere MQ poskytuje prostředky pro volání webových služeb.

Modul listener protokolu SOAP produktu WebSphere MQ podporuje služby hostované aplikací .NET Framework 1, .NET Framework 2 a Axis 1.4. Odesílatel SOAP produktu WebSphere MQ podporuje klienty webových služeb spuštěné v prostředí .NET Framework 1, .NET Framework 2, Axis 1.4 a Axis2. Klienti mohou být buď server WebSphere MQ , nebo klientská aplikace. Přenos produktu IBM WebSphere MQ pro SOAP je také integrován do jiných prostředí, jako je například Microsoft Windows Communication Foundation, WebSphere Application Server a CICS Transaction Server.

Integrace do produktu Microsoft Windows Communication Foundation je součástí podpory produktu IBM WebSphere MQ pro platformu .NET Framework 3.

Přenos IBM WebSphere MQ pro SOAP je sada protokolů a nástrojů pro transport zpráv SOAP pomocí JMS přes IBM WebSphere MQ. Je zabalen různými způsoby pro různá prostředí aplikace, jak je uvedeno v Tabulka 137 na stránce 915.

Tabulka 137. Přenos produktu IBM WebSphere MQ pro prostředí aplikace SOAP		
	Integrace s dalšími komponentami produktu WebSphere MQ	**Integrovaná do rámce**
Poskytnuto jako součást instalace produktu WebSphere MQ	.NET Framework 1 .NET Framework 2 Axis 1.4	Windows Communication Foundation (.NET Framework 3) Axis2 (pouze klient)
Poskytnuto v jiném softwarovém balíku		WebSphere Application Server CICS Transaction Server 4.1 WebSphere ESB WebSphere Process Server for Multiplatforms

Integrace přenosu IBM WebSphere MQ pro SOAP do aplikačního rámce zjednodušuje vývoj a implementaci webových služeb pro produkt IBM WebSphere MQ.

S dalšími komponentami SOAP produktu IBM WebSphere MQ můžete pracovat přímo s komponentami WebSphere SOAP MQ SOAP a vyvíjet a implementovat služby. Pomocí nástrojů IBM WebSphere MQ SOAP můžete konfigurovat a implementovat webové služby a klienty webové služby do produktu IBM WebSphere MQ.

V integrovaných prostředích je vývoj a implementace jednodušší. Stejné nástroje použijete pro vývoj a implementaci, jako byste mohli vyvinout a implementovat webovou službu SOAP HTTP. Stále je třeba konfigurovat fronty, kanály a správce front produktu IBM WebSphere MQ , které vyžadujete pomocí nástrojů WebSphere MQ .

Můžete mixovat a porovnávat klienty a servery SOAP IBM WebSphere MQ z libovolného z těchto prostředí.

Výhody

Přenos WebSphere MQ pro SOAP nabízí existujícím uživatelům produktu IBM WebSphere MQ následující hlavní výhody:

Pomocí sítě IBM WebSphere MQ můžete připojit existující webové služby.

Služby mohou být ty, které jste napsali, nebo služby poskytované jako rozhraní k jiným sbaleným softwarovým aplikacím, které jste implementovali.

Přínos pochází z použití vaší stávající sítě WebSphere MQ pro připojení webových služeb. Přenos produktu IBM WebSphere MQ má tu výhodu, že se jedná o spravovanou a spolehlivou službu systému zpráv ve frontě.

Zápis nových aplikací nebo převod existujících aplikací tak, aby místo rozhraní produktu IBM WebSphere MQ používal protokol SOAP.

Obvykle aplikace vyžadují, aby byl vyvinut specifický adaptér WebSphere MQ pro integraci s jinou aplikací. Adaptéry mají dvě části: část konektoru, která vkládá a získává zprávy do a z přenosu, a část adaptéru, která převádí data do a ze specifických formátů aplikace. Integrace jednotlivých párů aplikací je nová výzva.

Výhodou protokolu SOAP je standardizace protokolu SOAP pro definování aplikačních rozhraní a poté výběr transportů. Nemusíte psát specifické aplikační adaptéry a vy můžete zvolit, zda má být jako konektor použit IBM WebSphere MQ nebo HTTP. Kterou dopravu si zvolíte, záleží na tom, jaké kvality služeb a připojení požadujete.

Pro existující použití protokolu SOAP prostřednictvím protokolu HTTP je přínosem přenosu produktu WebSphere MQ pro protokol SOAP s použitím spravované a spolehlivé asynchronního přenosu. Výhody jsou dvojí:

Skutečně asynchronní programovací model pro dostupnost a výkon.
Pomocí asynchronního rozhraní klienta nemusí být klient a aplikace služeb k dispozici ve stejnou dobu. Požadavky odeslané klientem budou uloženy až do doby, kdy bude služba k dispozici pro zpracování.

Připravená sestavená spravovaná síť, která je navržena tak, aby byla spolehlivá a dostupná.

Vyberete-li volbu IBM WebSphere MQ jako transport, budete využívat výhod používání spravované sítě, která poskytuje spolehlivý systém zpráv.

Naproti tomu přenosy, jako jsou HTTP a FTP přes TCP/IP, jsou nespravované. Nespravovaná síť je ideální pro nepředvídatelné připojení: je zde méně úloh správy.

Souhrn

Přenos produktu IBM WebSphere MQ pro SOAP poskytuje následující komponenty:

- Transportní vazba SOAP/JMS se používá v dokumentech WSDL k vazbě služby SOAP s přenosem JMS. Implementace vazby SOAP/JMS v produktu WebSphere MQ používá identifikátor URI, který přijímá některou z následujících dvou formátů:

  Přenos WebSphere MQ pro SOAP

  ```
  jms:/queue?&Name=Value&Name=Value...
  ```

  Formát spoje WebSphere MQ pro doporučení kandidáta W3C

  ```
  jms:queue:qName?connectionFactory=connectQueueManager(qMgrName)&Name=Value&Name=Value...
  ```

- Mapování zprávy SOAP na zprávu produktu WebSphere MQ .
- Dva moduly listener SOAP IBM WebSphere MQ pro příjem požadavků SOAP, jeden pro jazyk Java a jeden pro prostředí .NET Framework 1 nebo .NET Framework 2. Listenery používají rozhraní .NET nebo Axis 1.4 ke zpracování požadavku SOAP.
- Dva odesilatelé SOAP IBM WebSphere MQ , aby vytvořili požadavky SOAP IBM WebSphere MQ . Klienti webových služeb se registrují s odesilatelem pro zpracování požadavků SOAP `jms:` .
- Integrace s produktem Windows Communication Foundation (WCF), někdy známá jako .NET 3, pro odesílání a příjem přenosu zpráv protokolu SOAP produktu WebSphere MQ .
- Integrace klienta s prostředím Axis2, někdy označovaná také jako rozhraní JAX-WS, k odeslání přenosu produktu WebSphere MQ Transport pro SOAP nebo W3C SOAP JMS.
- Příkaz **amqwdeployWMQService**, který vytváří komponenty pro vývoj a běhové komponenty a skripty k implementaci webové služby pomocí přenosu protokolu SOAP produktu IBM WebSphere MQ .
- Ukázkový klient Java a .NET a kód služby.
- Skript pro nastavení cesty ke třídě a ostatní skripty obslužných programů.

V integrovaných prostředích jsou odesilatel a modul listener integrováni do každého prostředí, stejně jako rozšíření nástrojů pro vývoj a implementaci.

Integrace protokolu SOAP a produktu WebSphere MQ

Přenos protokolu WebSphere MQ pro protokol SOAP rozšiřuje nástroje SOAP a nástroje webových služeb a běhové prostředí s produktem WebSphere MQ jako alternativním přenosem pro protokol HTTP pro protokol SOAP. Nemusíte upravovat existující webové služby pro použití přenosu produktu WebSphere MQ pro protokol SOAP jako přenosu. Přenos používá vlastní formát identifikátoru URI pro SOAP/JMS. Formát identifikátoru URI W3C pro SOAP/JMS je podporován omezeným způsobem klienty Axis2 .

Další řádek kódu musí být přidán do klientů v prostředích .NET Framework 1, .NET Framework 2 a Axis 1.4 . V klientech Axis 2 a Windows Communication Foundation (WCF) není třeba žádný další kód. Modul listener protokolu SOAP produktu WebSphere MQ spouští služby v prostředích .NET Framework 1, .NET Framework 2 a Axis 1.4 . Přenos produktu WebSphere MQ pro SOAP je integrován do některých dalších prostředí aplikačního serveru, včetně serverů WCF, CICS a WebSphere Application Server.

Co je SOAP?

Protokol SOAP[9]popisuje standardizovaný formát zpráv a protokolů interakce, které aplikace používají k výměně požadavků, odpovědí a datagramů. Protokol SOAP je nezávislý na přenosu, který se používá k přenosu zpráv, a prostředí aplikace, které odesílá a přijímá zprávy. W3C definuje SOAP verze 1.2 stručně:

Verze SOAP 1.2 poskytuje definici informací na bázi XML, které lze použít pro výměnu strukturovaných a typových informací mezi rovnocennými partnery v decentralizovaném, distribuovaném prostředí.[10].

Chcete-li použít SOAP, musí být svázán s přenosem, jako je například HTTP, e-mail nebo WebSphere MQ.

Rámec vazeb protokolu SOAP je sada pravidel pro přenos zprávy SOAP nad jiným protokolem, jako je například HTTP. SOAP verze 1.2 Část 2: Adjuncts (Second Edition) popisuje vazbu SOAP HTTP.

Doporučení pro kandidáta W3C , 4. června 2009, SOAP over Java Message Service 1.0, popisuje doporučení pro vazbu SOAP JMS. Vzhledem k tomu, že JMS je specifikace rozhraní API a nikoli přenosový protokol, doporučení SOAP JMS nepopisuje formát spojení se zprávami SOAP JMS. Popisuje protokoly interakce SOAP a vazbu rozhraní JMS API. V důsledku toho při použití doporučení SOAP JMS musíte i nadále používat stejnou implementaci platformy JMS pro klienta SOAP a server SOAP. Umožňuje spustit aplikaci SOAP JMS na libovolné implementaci platformy JMS. Implementace platformy JMS může být připojena k aplikačnímu serveru J2EE , pokud server i implementace rozhraní JMS vyhovují specifikaci JCA. Produkt WebSphere MQ JMS splňuje specifikaci JCA a lze jej připojit ke kompatibilním aplikačním serverům.

Přenos WebSphere MQ pro vazbu SOAP je stejně jako navrhovaný standard W3C , ale není to stejný. Jeho použití je popsáno v tématu Nastavení protokolu SOAPMQRFH2. Na rozdíl od doporučení kandidáta W3C není vazba SOAP formálně uvedena. Efektivně je to vazba HTTP a adresa služby má formu `jms:/queue?name=value&name=value...`, spíše než `http://authority/path?query#fragment`. `jms:` není oficiálně registrovaný schéma identifikátoru URI IANA.

Co je webová služba?

Protokol SOAP umožňuje, aby programy napsané v různých jazycích, které jsou spuštěny na různých platformách, komunikovaly pomocí různých přenosových protokolů. SOAP je specifikace protokolu. Webová služba je aplikace, která poskytuje službu prostřednictvím rozhraní SOAP, ke kterému lze přistupovat prostřednictvím internetových protokolů.

Důležitým cílem protokolu SOAP je poskytovat služby, které klienti mohou používat snadno. Jakmile jste navrhli, aby klient používal službu, můžete naprogramovat volání k vyvolání služby bez odkazu na externí dokumentaci. Rozhraní služby jsou popsána v XML v dokumentu WSDL. Dotaz, `http://authority/path?wsdl`, vrací popis WSDL služby SOAP.

Tip: Když implementujete webovou službu pro použití produktu WebSphere MQ, implementujte také službu do protokolu HTTP tak, aby standardní dotaz WSDL pracoval.

Vývoj webových služeb

Webové služby mají klienta a část služby. Služba se zapíše jako první, buď začíná od popisu rozhraní ve WSDL, nebo podle pravidel pro zápis třídy služeb. Sady nástrojů webových služeb mají obslužné programy pro generování kódu WSDL z definice rozhraní třídy; například **`java2wsdl`** nebo **`disco`**. Mají také nástroje pro generování nebo třídy_třídy z popisů rozhraní WSDL, například **`wsdl2java`**, **`wsimport`**nebo **`wsdl`**. První je známý jako vývoj zdola nahoru, a druhý shora dolů.

Příkaz **`amqwdeployWMQService`** v transportu produktu WebSphere MQ pro protokol SOAP používá tyto nástroje ke generování souborů WSDL, stubů klienta a serverů proxy klienta.

Webové služby jsou obvykle napsány pomocí integrovaného vývojového prostředí zaměřeného na konkrétní prostředí aplikačního serveru:

[9] Historicky se zkratka stala pro protokol SOAP (Simple Object Access Protocol).
[10] W3C: SOAP verze 1.2 -část 0

Vývojáře Eclipse IDE for Java EE
Vytvoří webové služby pro osu 2. Podporuje JAX-RPC a JAX-WS

Produkt Rational Application Developer V7.5 .
Vytvoří webové služby pro produkt WebSphere Application Server V7 a předchozí verze a také pro Axis. Podporuje JAX-RPC a JAX-WS.

WebSphere Integration Developer V6.2
Vytvoří webové služby pro produkt WebSphere Process Server a WebSphere ESB. Podporuje JAX-RPC a JAX-WS.

Visual Studio 2008 (verze 9)
Vytvoří webové služby pro .NET Framework 3.5 a starší (Windows Communication Foundation)

Visual Studio 2005 (verze 8)
Vytvoří webové služby pro .NET Framework 2 a starší

V kombinaci s přenosem WebSphere MQ pro protokol SOAP můžete použít kterýkoli z těchto nástrojů. Jakmile jste vyvinuli službu pro použití s HTTP, použijte nástroj **amqwdeployWMQService** k implementaci služeb pro použití produktu WebSphere MQ jako přenosu. Pomocí výstupu z nástroje můžete vytvořit nového klienta nebo upravit existující klienty tak, aby pro protokol SOAP používali přenos WebSphere MQ .

Je-li přenos WebSphere MQ pro SOAP integrován do prostředí aplikace, nemusíte používat nástroj **amqwdeployWMQService** nebo můžete upravit kód klienta. Vrstva SOAP klienta směruje požadavky klientů, které mají identifikátor URI s předponou `jms:` , do přenosu WebSphere MQ pro SOAP. Vrstva protokolu SOAP serveru volá přenos WebSphere MQ pro čekání na požadavky protokolu SOAP produktu `jms:` a vrací odpovědi na přenos WebSphere MQ pro SOAP.

Služby .NET se obvykle vyvinuly zdola nahoru pomocí anotací webových služeb v kódu a služeb Java shora dolů, pomocí definic rozhraní WSDL. Rozdíl v přístupech se zmenšuje, protože Java Standard Edition verze 6 podporuje rozhraní JAX-WS 2.0a používá anotace ke kvalifikaci definice rozhraní služeb. Nyní je snadné vyvinout služby Java zdola nahoru jako shora dolů. Který přístup zvolíte je otázka metody vývoje.

Klient webových služeb je zapsán po službě pomocí definice služby WSDL a generovaných stubů klienta a serverů proxy. V některých aplikacích není definice služby známá, když je klient zapsán. Klient načte WSDL služby a vytvoří požadavky na službu dynamicky. Obecněji je známá definice služby, ale adresa, na kterou je služba implementována, není známa. Sada nástrojů webové služby generuje rozhraní pro klienta, který se má použít k provádění požadavků na službu. Klient poskytuje adresu služby, je-li požadována. Ve třetím případě obsahuje WSDL všechny informace, které klient potřebuje. WSDL obsahuje jak rozhraní, tak adresu služby. Kód generovaný sadou nástrojů webové služby má všechny informace, které klient potřebuje k provádění požadavků na službu.

Pro SOAP v produktu WebSphere MQ můžete použít kterýkoli z těchto tří stylů.

Aplikační prostředí webových služeb

Sady nástrojů webových služeb vyžadují mapování z definice WSDL služby na proudy bajtů, které jsou přeneseny v požadavcích a odezvách SOAP. Bajtový proud je definován specifikací protokolu SOAP a je obsažen v obálce SOAP. Obálka SOAP je zobrazena v části Obrázek 166 na stránce 918.

```
<?xml version='1.0'?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Header> <!-- optional -->
<!-- headers... -->
</soap:Header>
<soap:Body>
<!-- payload or fault message -->
</soap:Body>
</soap:Envelope>
```

Obrázek 166. obálka SOAP

Mapování z obálky SOAP na vazbu jazyka a zpět je standardizovaný a částečně autorizovaný. Mapování je základní pro architekturu .NET a je poskytováno jako součást běhového prostředí Common Language Runtime (CLR). Mapování je standardizováno v jazyce Java podle specifikací JAX. Protože jsou mapování Java standardizována, klienti a služby webových služeb Java jsou přenositelné mezi různými aplikačními

prostředími Java. Rozhraní JAX-RPC (někdy nazývané JAX-WS 1.0) je mapování, které se nejvíce používá dnes. Je podporován Axis 1.4. JAX-WS (někdy označovaný jako JAX-WS 2.0) je výrazně vylepšený standard a pravděpodobně bude rychle nahrazovat JAX-RPC. JAX-WS je podporován Axis 2.0. Produkt WebSphere MQ 7.0.1 nepodporuje rozhraní JAX-WS a Axis 2.

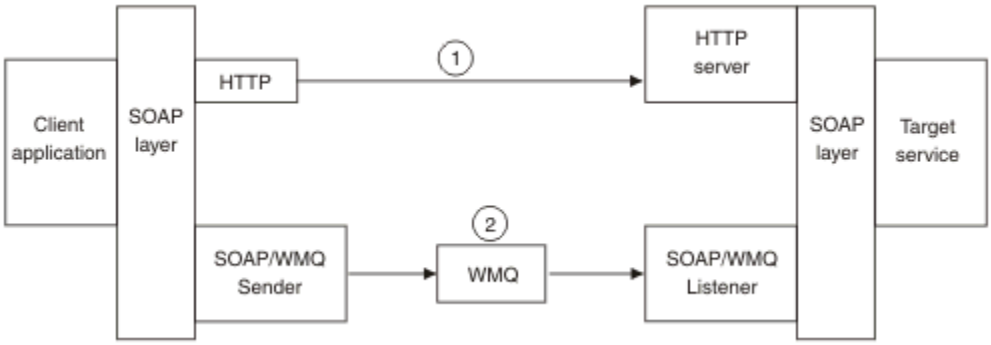
Přenos WebSphere MQ pro protokol SOAP nemění obsah obálky SOAP a obsah nemá vliv na přenos. Vazby jazyka mají vliv na přenos WebSphere MQ pro SOAP. Produkt WebSphere MQ 7.0.1 podporuje prostředí .NET Framework 1, .NET Framework 2 a Axis 1.4 s použitím kódu a obslužných programů dodávaných s přenosem WebSphere MQ pro protokol SOAP. Podpora pro přenos WebSphere pro protokol SOAP v prostředí .NET Framework 3 a 3.5 je implementována pomocí vlastního kanálu produktu WebSphere MQ pro Windows Communication Foundation.

Další vývojová prostředí SOAP a běhová prostředí mohou poskytovat podporu pro přenos WebSphere MQ pro protokol SOAP a podporovat různé jazyky. Například webové služby spuštěné na CICS podporují jazyky, jako jsou COBOL a PL/1.

Poznámka: Použité mapování nečiní žádný rozdíl v interoperabilitě webových služeb. Můžete mixovat a porovnávat klienty a služby napsané pomocí mapování .NET, JAX-RPC a JAX-WS.

Co je to přenos WebSphere MQ pro SOAP?

Přenos protokolu SOAP produktu WebSphere MQ je vazba SOAP a sada nástrojů webových služeb. Společně umožňují aplikacím vyměňovat zprávy SOAP s použitím produktu WebSphere MQ namísto HTTP. Obrázek 167 na stránce 919 zobrazí produkt WebSphere MQ jako alternativu k protokolu HTTP jako transport SOAP.

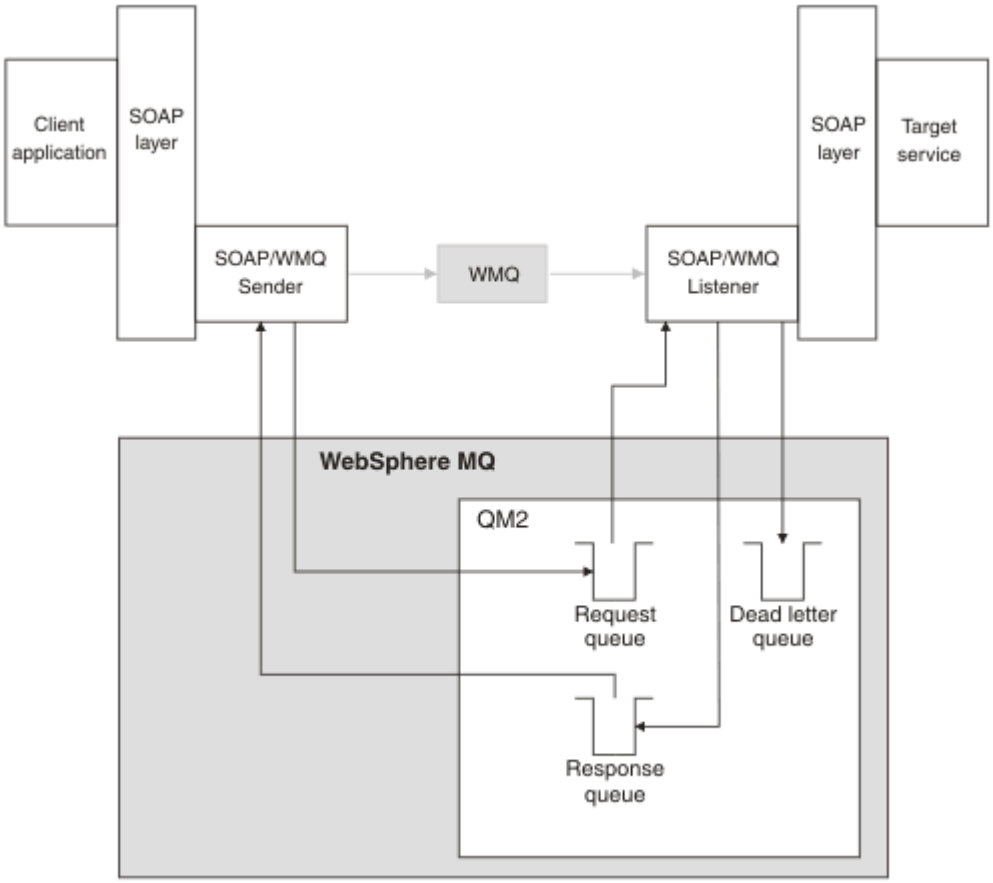


Obrázek 167. Přehled přenosu produktu WebSphere MQ pro protokol SOAP

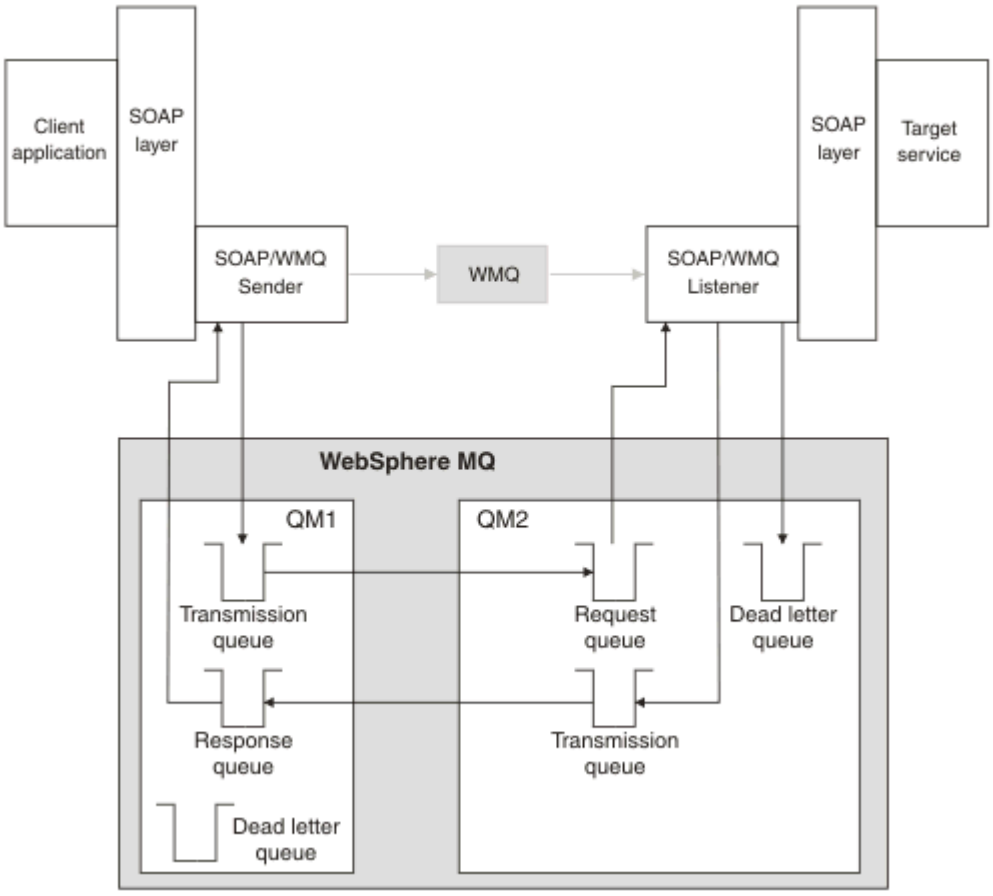
Protokol SOAP prostřednictvím protokolu HTTP se v diagramu zobrazuje jako (1). Vrstva SOAP klienta převádí požadavek do zprávy SOAP a komponenta HTTP posílá přes protokol TCP/IP. Komponenta serveru HTTP naslouchá požadavkům HTTP, obvykle na portu TCP/IP 80. Je-li požadavek pro službu SOAP, volá komponenta serveru HTTP vrstvu SOAP, aby převedla požadavek SOAP na volání metody. Poté vrátí odezvu.

Protokol SOAP prostřednictvím produktu WebSphere MQ je zobrazen jako (2). Aplikace typu klient registruje komponentu odesilatele WebSphere MQ SOAP jako obslužnou rutinu pro protokol `jms:` se vrstvou SOAP. Vrstva SOAP předává zprávy SOAP adresované odesilateli `jms:` produktu WebSphere MQ SOAP. Odesílatel používá identifikátor URI ve zprávě k umístění zprávy do fronty požadavků s použitím požadovaných kvalit služby. Odpovídající modul listener protokolu SOAP produktu WebSphere MQ čeká na zprávy ve své frontě požadavků a volá vrstvu SOAP za účelem zpracování požadavků a vrácení odpovědí.

Odesílatel protokolu SOAP a modul listener jsou normální programy produktu WebSphere MQ . Mohou být připojeny ke stejnému správci front jako v produktu Obrázek 168 na stránce 920nebo jsou připojeny k různým správcům front; viz Obrázek 169 na stránce 921. Klient může být připojen pomocí připojení klienta.



Obrázek 168. Fronty použité produktem SOAP/WebSphere MQ (jeden správce front)



Obrázek 169. Fronty použité produktem SOAP/WebSphere MQ (samostatné správce front)

Doporučení kandidáta W3C pro vazbu SOAP na JMS.

Doporučení kandidáta W3C definuje protokol SOAP nad vazbou služby JMS; Protokol SOAP nad Java Message Service 1.0. Také užitečné pro jeho příklady je Schéma identifikátoru URI pro službu Java(tm) Message Service 1.0[11].

Některé struktury aplikací, jako např. WebSphere Application Server v7, mají podporu pro doporučení kandidáta W3C . Odeslat požadavky SOAP formátované pomocí identifikátoru URI kompatibilního s doporučujícím doporučením W3C pomocí klienta Axis2 ; viz W3C SOAP over JMS URI for the WebSphere MQ Axis 2 client . Klient Axis2 odešle požadavek SOAP formátovaný buď pomocí přenosu W3C , nebo pomocí přenosu WebSphere MQ pro SOAP na základě identifikátoru URI v požadavku SOAP.

Podpora klienta Axis2 pro doporučení W3C je uvedena v opravné sadě 7.0.1.3 . Podpora pro jiné klienty a pro moduly listener SOAP poskytované produktem WebSphere MQ není poskytována.

Související pojmy

Implementace přenosu WebSphere pro SOAP na platformě .NET Framework 1, .NET 2 a Axis 1.4
Je možné, že budete chtít napsat vlastní odesilatele a modul listener produktu WebSphere MQ . Použijte implementaci produktu WebSphere MQ pro protokol SOAP v prostředí .NET Framework 1, .NET Framework 2 a Axis 1.4 jako vodítko.

Přenos produktu WebSphere MQ pro spolehlivý systém zpráv SOAP a webových služeb
Spolehlivý systém zpráv webových služeb je protokol, který spolehlivě vyměňuje požadavky webových služeb a odpovědi na nespolehlivé připojení. Je nejvhodnější pro řešení problémů přerušení spojení s krátkou životností.

11 Vyhledejte *Schéma identifikátoru URI pro rozhraní JMS* v odkazech specifikace W3C pro nejnovější koncept.

Implementace přenosu WebSphere pro SOAP na platformě .NET Framework 1, .NET 2 a Axis 1.4

Je možné, že budete chtít napsat vlastní odesilatele a modul listener produktu WebSphere MQ . Použijte implementaci produktu WebSphere MQ pro protokol SOAP v prostředí .NET Framework 1, .NET Framework 2 a Axis 1.4 jako vodítko.

1. Klientský program používá příslušný rámec webových služeb stejným způsobem jako pro transport HTTP. Musí také zaregistrovat předponu `jms:` . Předpona se zaregistruje buď pomocí metody jazyka Java produktu `com.ibm.mq.soap.Register.extension()` , nebo pomocí metody `IBM.WMQSOAP.Register.Extension()` CLR.
2. Rámec Axis 1.4 nebo .NET Framework 1 nebo 2 shrne volání do zprávy požadavku SOAP přesně jako pro SOAP/HTTP.
3. Služba WebSphere MQ je identifikována identifikátorem URI s předponou `jms:`. Pokud rámec identifikuje identifikátor URI produktu `jms:` , zavolá kód odesílatele přenosu produktu WebSphere MQ ; `com.ibm.mq.soap.transport.jms.WMQSender` (pro Axis 1.4) nebo `IBM.WMQSOAP.MQWebRequest` (pro .NET1 a 2). Pokud rámec narazí na identifikátor URI s předponou `http:` , zavolá odesilatele protokolu SOAP přes HTTP.
4. Zpráva SOAP je přenášena odesilatelem SOAP produktu WebSphere MQ pomocí fronty požadavků. **`SimpleJavaListener`** (pro Java) nebo **`amqwSOAPNETListener`** (pro .NET) přijímá zprávu požadavku.

   Moduly listener protokolu SOAP produktu WebSphere MQ jsou samostatné procesy a jsou s podporou podprocesů s přizpůsobným počtem podprocesů.
5. Modul listener protokolu SOAP produktu WebSphere MQ načte příchozí požadavek SOAP a předá jej příslušné infrastruktuře webových služeb.
6. Infrastruktura webové služby analyzuje zprávu požadavku SOAP a vyvolává službu. Procedura je stejná jako u zprávy, která dorazila na přenos HTTP.
7. Infrastruktura formátuje odezvu na zprávu odpovědi SOAP a vrací ji do modulu listener protokolu SOAP produktu WebSphere MQ .
8. Modul listener umístí zprávu do fronty odpovědí a zpráva se přenese do odesilatele WebSphere MQ SOAP. Odesílatel ji předá do infrastruktury webové služby klienta.
9. Infrastruktura klienta zanalyzuje zprávu odezvy SOAP a předá výsledek zpět aplikaci klienta.

Každý kontext aplikace je obsluhován samostatnou frontou požadavků produktu WebSphere MQ .

Kontext aplikace je řízen v rámci Axis 1.4 tím, že se zajistí, aby modul listener a služba protokolu SOAP produktu WebSphere MQ bylo provedeno v příslušném adresáři. Osa 1.4 nastavuje správnou proměnnou CLASSPATH pro daný adresář.

Kontext aplikace je řízen v prostředí .NET modulem listener protokolu SOAP produktu WebSphere MQ , který provádí službu v kontextu vytvořeném při volání do produktu `ApplicationHost.CreateApplicationHost`. Volání určuje cílový adresář provedení. Každá služba poté pracuje v adresáři, ve kterém byla implementována.

`amqwdeployWMQService` generuje fronty požadavků a požadavků. Vygeneruje také infrastrukturu potřebnou pro práci s frontami a implementaci služeb na Axis 1.4.

Související pojmy

Integrace protokolu SOAP a produktu WebSphere MQ

Přenos produktu WebSphere MQ pro spolehlivý systém zpráv SOAP a webových služeb
Spolehlivý systém zpráv webových služeb je protokol, který spolehlivě vyměňuje požadavky webových služeb a odpovědi na nespolehlivé připojení. Je nejvhodnější pro řešení problémů přerušení spojení s krátkou životností.

Přenos produktu WebSphere MQ pro spolehlivý systém zpráv SOAP a webových služeb

Spolehlivý systém zpráv webových služeb je protokol, který spolehlivě vyměňuje požadavky webových služeb a odpovědi na nespolehlivé připojení. Je nejvhodnější pro řešení problémů přerušení spojení s krátkou životností.

Produkt WebSphere MQ for SOAP využívá pro předání zpráv SOAP prostřednictvím spravované a spolehlivé sítě produktu WebSphere MQ . Přenosy jako HTTP a FTP jsou nespravované. Nespravované sítě jsou ideální pro nepředvídané připojení, kde potíže a náklady na správu připojení převažují nad výhodami neztrácející požadavky a odezvy.

Chcete-li překonat problém ztráty souborů při přerušení spojení v nespravovaných sítích, služby jako spravované FTP vytvářejí vrstvu správy na vrcholu FTP. Vrstva správy přebírá zátěž kontroly, že soubory byly úspěšně přeneseny od uživatelů, a v případě potřeby opakovaně přenáší chybějící soubory. Chcete-li používat spravovaný FTP, musíte mít nainstalovaný software pro správu na obou koncích připojení.

Spolehlivý systém zpráv webových služeb (WSRM) používá odlišný přístup k řešení problému nespolehlivých připojení. Jeho cílem je spolehlivě přenášet požadavky a odpovědi webových služeb, aniž by oba konce připojení museli používat stejný software. Jakýkoli software, který implementuje protokol spolehlivého systému zpráv webových služeb, může spolehlivě vyměňovat zprávy s jiným softwarem.

Dojde-li k selhání připojení, musí odesilatel a příjemce zachovat kontext přenosu zpráv WSRM s použitím generovaného identifikátoru URI jako klíče. Odesílatel a příjemce se neustále pokouší o vytvoření nového připojení. Je-li úspěšně ustanoveno nové připojení, přenos se dokončí. Specifikace WSRM neuvádí, jak je kontext zachován, nebo když se pokusíte o nové připojení.

Můžete se rozhodnout, že se zajímají pouze výpadky s krátkou životností. V případě delších výpadků můžete být připraveni zrušit přenosy, které nebylo možné po určité době restartovat. Podobně můžete být připraveni vyřadit přenosy v případě, že selže klient nebo služba. Ponechání uživatele zodpovědného za zajišťování přenosů, klade méně požadavků na správu koordinace klienta a služby.

Pokud jsou výpadky sítě typu long-žili déle než 30 minut nebo pokud dojde k selhání klienta nebo serveru, je zde zvýšená pravděpodobnost, že některá připojení nebudou nikdy znovu zavedena. Nyní již nelze spoléhat na to, že služba WSRM bude automaticky obnovovat přenos zpráv v nespravovaném způsobu. Je třeba zvážit správu nezdařených připojení WSRM, což znamená vývoj softwaru pro správu sítě klientů a služeb.

Použití WSRM k překonání krátkých výpadků by mohlo výrazně snížit práci se ztrátou zpráv v mobilní síti. Pokud nebudete muset zajistit doručování zpráv, mohou výhody snížení ztráty zpráv ospravedlnit další náklady na vývoj implementace WSRM.

Protokol SOAP prostřednictvím rozhraní JMS poskytuje zajištěné doručování zpráv a zabývá se delší dobou trvání výpadků klienta, serveru a sítě. Pokud hledáte spolehlivější kvalitu služby pro SOAP než HTTP, které řešení zvolíte: WebSphere MQ transport pro SOAP nebo WSRM? Odpověď závisí na mnoha faktorech. Některé z faktorů, které je třeba zvážit, jsou uvedeny níže:

1. Určuje, zda je nespolehlivost způsobena selháním připojení.
2. Jak dlouho mají nemilovaná selhání spojení.
3. Pokud můžete spravovat jak straně klienta, tak i straně serveru připojení.
4. Pokud je uživatel nebo administrátor v konečném důsledku odpovědný za doručování zpráv.

Související pojmy

Integrace protokolu SOAP a produktu WebSphere MQ

Implementace přenosu WebSphere pro SOAP na platformě .NET Framework 1, .NET 2 a Axis 1.4
Je možné, že budete chtít napsat vlastní odesilatele a modul listener produktu WebSphere MQ . Použijte implementaci produktu WebSphere MQ pro protokol SOAP v prostředí .NET Framework 1, .NET Framework 2 a Axis 1.4 jako vodítko.

Instalace a ověření webových služeb produktu WebSphere MQ

Použijte pokyny v těchto tématech k instalaci a ověření přenosu WebSphere MQ pro SOAP.

Instalace produktu WebSphere MQ Web transport pro SOAP

Tyto pokyny použijte k instalaci webového přenosu produktu WebSphere MQ pro SOAP. Instalace vytváří nástroje pro spouštění klientů a služeb webových služeb pomocí produktu WebSphere MQ jako přenosu protokolu SOAP. Nástroje se používají v prostředích .NET Framework 1, .NET 2, Axis 1.4nebo Axis2 SOAP.

Než začnete

Zkontrolujte předpokládané produkty na adrese Systémové požadavky pro produkt IBM WebSphere MQ. Proces instalace nekontroluje přítomnost a dostupnost předem vyžadovaného softwaru. Musíte ověřit, zda jsou předpoklady nainstalovány.

Produkt WebSphere MQ poskytuje kopii běhového prostředí Axis 1.4 . Použijte tuto verzi s produktem WebSphere MQ namísto jiného, který jste mohli instalovat. IBM neposkytuje technickou podporu pro Apache Axis. Kontaktujte Softwarovou nadaci Apache , pokud s ní máte technické problémy.

Chcete-li spustit webové služby v prostředí .NET Framework 3 SOAP, produkt WebSphere MQ používá Windows Communication Foundation. Vlastní kanál produktu WebSphere MQ pro Windows Communication Foundation spouští klienty a služby webových služeb pomocí produktu WebSphere MQ jako přenosu pro zprávy SOAP.

Informace o této úloze

Webový transport produktu WebSphere MQ pro protokol SOAP můžete nainstalovat buď jako aplikaci klienta WebSphere MQ MQI, nebo jako aplikaci serveru. Pokud jste již produkt WebSphere MQ instalovali jako klienta nebo server v počítači, zkontrolujte, zda jste nainstalovali uvedené komponenty.

Produkt *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, do kterého je produkt WebSphere MQ nainstalován.

Proveďte následující kroky instalace.

Postup

1. Vyberte komponentu "Java a. Net Messaging and Web services" pro instalaci.
2. V systémech Solaris a HP-UXvyberte pro instalaci komponentu "Java Runtime Environment" .
3. Vyberte sadu vývojových nástrojů pro instalaci.
4. Nainstalujte a ověřte produkt WebSphere MQ , jak je popsáno v tématu Quick Beginning for your platform.
5. Okopírujte běhové prostředí Apache Axis 1.4 , `axis.jar` z adresáře `prereqs/axis` na instalačním médiu produktu WebSphere MQ . Zkopírujte jej do instalačního adresáře popsaného v části Tabulka 138 na stránce 925, Tabulka 139 na stránce 925 nebo Tabulka 140 na stránce 925.

   Windows

   ```
   Copy D:\PreReqs\axis\axis.jar MQ_INSTALLATION_PATH\java\lib\soap
   ```

   AIX

   ```
   cp -f PreReqs/axis/axis.jar MQ_INSTALLATION_PATH/java/lib/soap/axis.jar
   chown mqm:mqm MQ_INSTALLATION_PATH/java/lib/soap/axis.jar
   chmod 444 MQ_INSTALLATION_PATH/java/lib/soap/axis.jar
   ```

   Instalační adresáře HP-UX, Solaris a Linux (všechny platformy)

   ```
   cp -f PreReqs/axis/axis.jar MQ_INSTALLATION_PATH/java/lib/soap/axis.jar
   chown mqm:mqm MQ_INSTALLATION_PATH/java/lib/soap/axis.jar
   chmod 444 MQ_INSTALLATION_PATH/java/lib/soap/axis.jar
   ```

6. V systému Windows 2003 spusťte **Aspnet_regiis.exe** pro aktualizaci map skriptů tak, aby ukazovaly na verzi běhového prostředí Common Language Run, kterou používáte.

   Hledejte obslužný program **Aspnet_regiis.exe** v produktu %SystemRoot%Microsoft.NET/Framework/*version-number* .

7. Nastavte proměnnou prostředí `WMQSOAP_HOME` tak, aby ukazovala na instalační adresář produktu WebSphere MQ .

Výsledky

Tabulka 138. Instalační adresáře systému Windows	
Umístění	**Obsah**
MQ_INSTALLATION_PATH\programs\bin	Binární, příkaz, DLL a spustitelné soubory
MQ_INSTALLATION_PATH\programs\java\lib	`.jar files`
MQ_INSTALLATION_PATH\programs\java\lib\soap	Protokol SOAP `.jar files`
MQ_INSTALLATION_PATH\programs\soap\samples	Vzorky a IVT

Tabulka 139. Instalační adresáře AIX	
Umístění	**Obsah**
MQ_INSTALLATION_PATH/bin	Skripty shell
MQ_INSTALLATION_PATH/java/lib	`.jar files`
MQ_INSTALLATION_PATH/java/lib/soap	Soubory `axis.jar` a další soubory `.jar` JAX-RPC
MQ_INSTALLATION_PATH/samp/soap	Vzorky a IVT

Tabulka 140. Instalační adresáře HP-UX, Solaris a Linux (všechny platformy)	
Umístění	**Obsah**
MQ_INSTALLATION_PATH/bin	Skripty shell
MQ_INSTALLATION_PATH/java/lib	`.jar files`
MQ_INSTALLATION_PATH/java/lib/soap	Soubory `axis.jar` a další soubory `.jar` JAX-RPC
MQ_INSTALLATION_PATH/samp/soap	Vzorky a IVT

Jak pokračovat dále

1. Pouze pro prostředí .NET, musíte registrovat přenos WebSphere MQ pro soubory SOAP s globální mezipamětí sestavení. Pokud je produkt .NET již nainstalován při instalaci produktu WebSphere MQ, registrace se provádí automaticky při instalaci. Pokud nainstalujete prostředí .NET po WebSphere MQ, registrace se provede automaticky, když se IVT poprvé spustí.

   Produkt **amqiregisterdotnet.cmd** můžete spustit, chcete-li provést registraci sestavení .NET. Můžete také spustit příkaz **amqiregisterdotnet.cmd** k vynucení opětovné registrace v libovolné fázi. Po provedení této registrace přežije restart systému a následná opětovná registrace není obvykle nutná.
2. Spusťte test ověření instalace, jak je popsáno v tématu “Ověření transportu produktu IBM WebSphere MQ pro protokol SOAP” na stránce 926.
3. Hodláte-li vyvíjet klienta Axis2 , musíte stáhnout Axis2 1.4.1 z Apache; viz “Vyvíjení klienta JAX-WS pro přenos WebSphere pro protokol SOAP pomocí prostředí Eclipse” na stránce 949.

Ověření transportu produktu IBM WebSphere MQ pro protokol SOAP

Ověřte přenos IBM WebSphere MQ pro protokol SOAP pomocí příkazu **runivt** . Příkaz spustí celou řadu demonstračních aplikací a zajistí, aby prostředí bylo po instalaci správně nastaveno.

Než začnete

Než spustíte příkaz **runivt** , ujistěte se, že máte následující běhová prostředí:

- Chcete-li spustit pouze na ose Axis, musíte mít ve svém systému k dispozici sadu Java SDK (v rámci specifikace SOE). Musíte také zahrnout umístění příkazů `java.exe` a `javac.exe` v systémové proměnné prostředí **PATH** .
- Chcete-li spustit test pouze na platformě .NET (podporováno pouze v systému Windows), musíte mít ve svém systému jak sadu nástrojů Java SDK, tak i kompilátory .NET a nástroje .NET. Chcete-li tak učinit, vstupte buď do příkazového řádku produktu Visual Studio, nebo do příkazového řádku Microsoft Windows SDK, a potom přidejte umístění souborů `java.exe` a `javac.exe` do proměnné prostředí **PATH** .
- Chcete-li spustit všechny dostupné testy: pro platformy Windows , musí být prostředí nakonfigurováno způsobem popsaným v testovacím běhu testu .NET. Na platformách UNIX and Linux musí být prostředí konfigurováno tak, jak je popsáno v testovacím běhu pouze Axis.

Informace o této úloze

Místo spuštění testu ověření na rozhraní .NET a Axis možná budete chtít spustit test pouze na ose Axis nebo pouze na platformě .NET.

Pokud se setkáte s problémy s testy a chcete znovu začít:

1. Zastavte správce front `WMQSOAP.DEMO.QM` pomocí volby `immediate` .
2. Zastavte modul listener, který byl spuštěn v jiném okně.
3. Odstraňte správce front.
4. Odstraňte adresář dočasných ukázek, který jste vytvořili a začněte znovu.

Na platformách UNIX and Linux je nutné spustit příkaz pomocí relace systému X Windows .

Příkaz **runivt** mění obsah adresáře `soap/samples` . Chcete-li uchovat obraz instalace nezměněný, zkopírujte adresář ukázek do dočasného umístění a spusťte verifikační test z dočasného umístění.

Verifikaci instalace můžete spustit tolikrát, kolikrát chcete.

Proveďte následující kroky, chcete-li ověřit instalaci přenosu produktu IBM WebSphere MQ pro protokol SOAP na rozhraní .NET Framework 1, .NET Framework 2 a Axis 1.4:

Postup

1. Zkopírujte adresářový strom `./tools/soap/samples` do dočasného umístění.
2. Spusťte příkazové okno s dočasným adresářem jako aktuální adresář.
3. Použijte příkaz **runivt** ke spuštění testu instalace. Skript `runivt` zkompiluje testovací třídu, ukázkového klienta a služby před implementací a spuštěním těchto tříd. Pro testovací třídu, ukázkový klient a služby ke spuštění dokončete kroky instalace uvedené v tématu Instalace produktu WebSphere(r) MQ Web Transport for SOAP a ujistěte se, že příkazový řádek použitý ke spuštění příkazu `runivt` má nastavenou běhovou sadu prostředí. Ke spuštění příkazu **runivt** použijte libovolnou z následujících metod:
   - Spustit test pouze na ose Axis: `runivt Axis`.
   - Spustit test pouze na platformě .NET (podporováno pouze v systému Windows): `runivt DotNet`.
   - Spusťte všechny dostupné testy:`runivt`.

   Další informace o syntaxi a parametrech příkazu `runivt` naleznete v příručce **runivt**: IBM WebSphere MQ transport for SOAP installation verification test. Testy, které můžete spustit, jsou

vypsány v souboru `ivttests.txt` na platformách Windows a `ivttests_unix.txt` na platformách UNIX and Linux .

Související odkazy

runivt: Test ověření instalace produktu WebSphere MQ pro instalaci SOAP

Vývoj webových služeb pro přenos WebSphere MQ pro SOAP

Použijte normální vývojové prostředí webové služby pro vývoj služeb pro použití s přenosem WebSphere MQ pro protokol SOAP.

Než začnete

1. Plánujete-li používat nástroje příkazového řádku dodávané s přenosem WebSphere MQ pro SOAP:
   a. Vytvořte adresář implementace pro službu.
   b. Spusťte příkazové okno v adresáři.
   c. Pro .NET, csc.exe a wsdl.exe musí být v cestě a musí být ze stejné verze prostředí .NET Framework.
   d. Pro Java,
      i) Spuštěním příkazu **amqwsetcp** nastavte cestu ke třídě.
      ii) Prostředí IBM JRE a sada JDK na stejné úrovni verze musí být v aktuální cestě. Úroveň verze musí být nejméně 5.0.
      iii) Upravte cestu ke třídě tak, aby obsahovala umístění dalších knihoven produktu `.jar` a adresářů obsahujících soubory .java, včetně balíků pro vývoj, který vyvíjíte. Umístěte aktuální adresář `"."` do cesty ke třídě.
      iv) Vytvořte adresář vzhledem k aktuálnímu adresáři v okně příkazového řádku, který odpovídá názvu balíku služby, kterou vyvíjíte.
2. Případně můžete použít nástroje pracovní plochy, které podporují vývoj webových služeb. Ukázkové úlohy vývoje používají produkt Microsoft Visual Studio 2008, Eclipse IDE for Java EE Developers and WebSphere Application Server Community Edition.

Informace o této úloze

Existující webové služby nepotřebují žádnou úpravu pro práci s přenosem WebSphere pro SOAP. Nástroje dodávané s přenosem WebSphere MQ pro implementaci SOAP implementují webovou službu a spouštějí jej s použitím modulu listener SOAP produktu WebSphere MQ . Nástroje také generují služby WSDL, stuby .NET klienta a třídy proxy produktu `.java` pro vývoj přenosu WebSphere MQ pro klienty SOAP.

Postupujte takto, chcete-li vytvořit službu a připravit ji na implementaci a generování klientů. Chcete-li vytvořit službu pomocí produktu Eclipse nebo Microsoft Visual Studio 2008, postupujte podle kroků souvisejících úloh.

Postup

1. Vyvinout službu pomocí běžného vývojového prostředí.
2. Testování služby pomocí klienta webových služeb HTTP
3. Chcete-li připravit adresář implementace, postupujte takto:
   - Pro prostředí Java
     a. Zkopírujte soubor `.java` , který definuje rozhraní služby, do adresáře implementace.
     b. Zkopírujte všechny soubory produktu `.class` pro danou službu do adresáře, který odpovídá názvu balíku.
     c. Zkontrolujte, zda cesta ke třídě může vyhledat všechny požadované třídy: zkompilujte soubor služby `.java` pomocí produktu **javac**.
   - Pro .NET

a. Copy the `.asmx` file defining the service into the deployment directory, and

b. Používáte-li model s kódem, zkopírujte všechny soubory `.dll` do adresáře *deployment directory*`\bin` .

Vývoj služby JAX-RPC pro přenos WebSphere MQ pro SOAP pomocí prostředí Eclipse

Vyvinout webovou službu Axis 1.4 , která má být spuštěna s použitím produktu WebSphere MQ jako poskytovatele služeb. Použijte normální prostředí pro vývoj webových služeb k vytvoření služby pro implementaci na Axis 1.4.

Než začnete

Zvemte v úvahu požadavky na implementaci webového serveru do modulu listener SOAP produktu WebSphere MQ pro osu 1.4.

- Modul listener protokolu SOAP produktu WebSphere MQ pro osu 1.4 vyžaduje prostředí IBM JRE ve verzi 5.0 nebo vyšší. Prostředí JRE a sada JDK použité pro vývoj musí být na stejné úrovni verze.
- Modul listener protokolu SOAP produktu WebSphere MQ pro osu 1.4 vyžaduje instalaci produktu `axis.jar` s produktem WebSphere MQ. Změňte cestu sestavení ve svém vývojovém prostředí tak, aby odkazovala na soubor `axis.jar` instalovaný s produktem WebSphere MQnamísto souborů produktu `axis.jar` nainstalovaných s vývojovým prostředím.
- Do produktu WebSphere MQ V7.0.1je kód WSDL vygenerovaný pro implementovanou službu RPC/kódovaný RPC/kódován. Ve verzi V7.1můžete také požadovat WSDL stylu RPC/literal. Generovaný WSDL se používá pouze pro implementaci. Službu můžete definovat pomocí WSDL kompatibilního s WS-I.

Informace o této úloze

Vytvořte službu pomocí běžného vývojového prostředí webových služeb.

V této úloze používáme volně dostupný prostředek Eclipse Java EE IDE for Web Developers, známý jako Galileo. Pro aplikační server používáme produkt WebSphere Application Server Community Edition v2.1 (Community Edition), založený na Geronimo. Informace o tom, jak získat, instalovat a konfigurovat prostředí IDE a server, najdete v souvisejících úlohách.

Otestujte službu pomocí protokolu HTTP jako přenosu s použitím průzkumníku webových služeb, který je poskytován v integrovaném vývojovém prostředí. Případně vygenerujte proxy klienta HTTP a otestujte službu pomocí svého vlastního kódu klienta.

Při vytváření webové služby zdola nahoru můžete postupovat podle těchto kroků. Jako příklad použijte ukázkový program `StockQuoteAxis.java` .

Postup

1. Spusťte produkt Eclipse IDE for Java EE Developers s novým pracovním prostorem.
2. Nakonfigurujte pracovní prostor tak, aby používal Java50,

   Produkt WebSphere Application Server Community Edition 2.1.4 nepracuje s uživatelem Java60.

   a) **Okno** > **Předvolby** > **Java** > **Instalovaná prostředí JRE** > **Přidat ...** > **Standardní VM** > **Další** > **Adresář ...**

   b) Přejděte do instalačního adresáře **Java50** > **OK** > **Dokončit** .

   c) Zkontrolujte prostředí JRE **Java50** > OK
3. Přidejte běhové prostředí Community Edition a spusťte produkt Community Edition.

   a) **Okno** > **Předvolby** > **Server** > **Běhová prostředí** > **Přidat ...**

   b) Vyberte volbu **IBM WASCE v2.1** ze seznamu **Nové běhové prostředí serveru** > zaškrtněte **Vytvořit nový lokální server** > **Další**

   Není-li produkt **IBM WASCE 2.1** v seznamu uveden, je třeba provést dvě další úlohy:

   i) Nainstalujte produkt WebSphere Application Server Community Edition.

ii) Nainstalujte aktualizaci Eclipse pro Community Edition.

Najděte si podrobnosti na adrese WebSphere Application Server Community Edition .

c) Přejděte do instalačního adresáře aplikačního serveru > **OK** > **Dokončit** > **OK**.

d) Klepněte pravým tlačítkem myši na volbu **IBM WASCE v2.1 server** v pohledu Servery > **Start** .

Tip: WASCE můžete spravovat v prostředí Eclipse: Right-click **IBM WASCE v2.1 server** > **Spustit konzolu WASCE** . Standardní **Username** a **Password** jsou `system` a `manager` .

4. Nastavení serveru a běhového prostředí pro webové služby.

a) **Okno** > **Předvolby** > **Webové služby** > **Server a běhové prostředí**

b) Vyberte volbu **IBM WASCE v2.1 Server** jako server.

c) Ponechte volbu **Apache Axis** jako běhovou komponentu webové služby.

5. Vytvořte dynamický webový projekt.

a) **Soubor** > **Nový** > **Dynamický webový projekt**.

Pojmenujte projekt `StockQuoteAxis`.

b) Zaškrtněte volbu **Přidat projekt do EAR** > **Nový ...**

c) Na stránce **Projekt aplikace EAR** zadejte **Project name** `StockQuoteAxisEAR` > **Dokončit** .

Odpovězte **OK** v odpovědi na dialogové okno s návrhem na přepnutí do perspektivy Java EE , nebo můžete zůstat v perspektivě Java, abyste tyto instrukce přesně následovali.

d) Jako cílové běhové prostředí je vybráno **serverIBM WASCE 2.1** . Přijměte jej a ostatní výchozí nastavení > **Dokončit**.

Odpovězte **OK** v odpovědi na dialogové okno s návrhem na přepnutí do perspektivy Java EE , nebo můžete zůstat v perspektivě Java, abyste tyto instrukce přesně následovali.

6. Import ukázkového programu `StockQuoteAxis.java`

a) Otevřete webový projekt **StockQuoteAxis** > Klepněte pravým tlačítkem myši na složku **src** > **Importovat ...**

b) Vyberte volbu **Obecné** > **Systém souborů** > **Další** .

c) Přejděte na adresu *MQ_INSTALLATION_PATH*`\tools\soap\samples\java\server` > zkontrolujte **StockQuoteAxis.java** > **Dokončit**

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, kde je nainstalován produkt WebSphere MQ . Chcete-li zobrazit soubory, které obsahuje, musíte zvýraznit adresář serveru.

7. Opravte chybu kompilace přesunutím `StockQuoteAxis.java` do jeho správného balíku.

a) Otevřete produkt `StockQuoteAxis.java` a klepněte pravým tlačítkem myši na problém > **Rychlá oprava** .

b) Poklepejte na volbu **Přesunout 'StockQuoteAxis.java' na balík 'soap.server'** > **Uložit**.

8. Vytvořte webovou službu z produktu `StockQuoteAxis.java` .

a) Klepněte pravým tlačítkem myši na nabídku `StockQuoteAxis.java` > **Webové služby** > **Vytvořit webovou službu** > **Další**.

Přijměte výchozí konfiguraci pro službu:

Typ webové služby
Zdola Java beanWeb Service

Implementace služby
soap.server.StockQuoteAxis

Server
`IBM WASCE v2.1 server`

Běhové prostředí webové služby
Osa Apache

projekt služby
Osa StockQuote

Projekt EAR služby
StockQuoteAxisEAR

Konfigurace
Žádná generace klienta

9. Vyberte metody pro přístup a styl webové služby > **Další**.

   Spusťte server, pokud jste k tomu vyzváni.

   a) Ponechte všechny vybrané metody.

   b) Vyberte styl document/literal (wrapped).

10. **Dokončit**

    Když je služba implementována, podívejte se do složky `WebContent\wsdl` ve webovém projektu `StockQuoteAxis` a najděte vygenerovaný soubor `StockQuoteAxis.wsdl` .

11. Otestujte službu pomocí protokolu HTTP s průzkumníkem webových služeb.

    a) Right-click `StockQuoteAxis.wsdl` > **Test with Web Services Explorer**.

    b) Klepněte na operaci **getQuote** v akcích **StockQuoteAxisSoapBinding** v okně **Průzkumník webových služeb** .

    c) Napište `ibm` do vstupního pole **symbol** > **Go** .

12. Otestujte službu pomocí transportu produktu WebSphere MQ pro protokol SOAP.

    Chcete-li implementovat službu, použijte **`SimpleJavaListener`**, která je v `com.ibm.mq.soap.jar`. Musíte přidat knihovny Java a SOAP produktu WebSphere MQ do cesty sestavení.

    a) Klepněte pravým tlačítkem myši na webový projekt **StockQuoteAxis** > **Cesta sestavení** > **Konfigurovat cestu sestavení ...**

    b) Klepněte na kartu **Knihovny** > **Přidat externí soubory Jar ...**. Přejděte na adresu *`MQ_INSTALLATION_PATH`*`\java\lib` . a vyberte všechny soubory `.jar` > **Otevřít** > **Přidat externí soubory JAR ...**. Přejděte na *`WMQ Install directory`*`\java\lib\soap` a vyberte všechny soubory `.jar` > **Otevřít** > **OK**.

    Produkt *`MQ_INSTALLATION_PATH`* představuje adresář vysoké úrovně, do kterého jste nainstalovali produkt WebSphere MQ .

    c) V Průzkumníku projektů klepněte pravým tlačítkem myši na nabídku `StockQuoteAxis\Java Resources\Libraries\com.ibm.mq.soap.jar\com.ibm.mq.soap.transport.jms\SimpleJavaListener.class\ SimpleJavaListener` > **Spustit jako ...**> **Spustit konfigurace ...**

    Tip:

    Pokud neexistuje žádná konfigurace produktu `SimpleJavaListener` , klepněte na ikonu **Nová konfigurace** na stránce **Vytvořit, spravovat a spustit konfigurace** průvodce **Spustit konfigurace** .

    Příkaz **`SimpleJavaListener`** nemá k dispozici žádný příkaz k jeho zastavení. Chcete-li monitorovat nebo zastavit produkt **`SimpleJavaListener`**, otevřete **perspektivu Ladění** v prostředí Eclipse.

    d) Otevřete kartu **(x) = Argumenty** . Do vstupní oblasti **Argumenty programu** zadejte parametry pro **`SimpleJavaListener`**.

    Pro tento příklad zadejte

    ```
    -u "jms:/queue?destination=REQUESTAXIS@QM1&connectionFactory=()
    &initialContextFactory=com.ibm.mq.jms.Nojndi
    &targetService=StockQuoteAxis
    &replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE" -n 10
    ```

    Poznámka: Cílová služba je `StockQuoteAxis` , aby odpovídala názvu cílové služby vytvořenému v deskriptoru implementace služby, `StockQuoteAxis\WebContent\WEB-INF\server-config.wsdd` . amqwdeployWMQService vytvoří cílovou službu s názvem

soap.server.StockQuoteAxis. V tomto příkladu používáte stejné StockQuoteAxis.class a service-config.wsdd jako server HTTP.

e) Na stejné kartě nakonfigurujte **Pracovní adresář** tak, aby odkazoval na soubor server-config.wsdd : ${workspace_loc:StockQuoteAxis/WebContent/WEB-INF}

a) **Spustit**

Chyby se zapisují na konzolu. Zůstane-li konzola prázdná, produkt **SimpleJavaListener** se spustí v pořádku.

b) Chcete-li otestovat nasazení, spusťte klienta Axis StockQuote, vyvinutý v úloze, “Vývoj klienta JAX-RPC pro transport WebSphere pro protokol SOAP pomocí prostředí Eclipse” na stránce 941.

Příklad: Ukázkový program Axis StockQuote

Vzorová webová služba Java, StockQuoteAxis.java, je instalována v *WMQ install directory*\tools\soap\samples\java\server . StockQuoteAxis.java, Obrázek 170 na stránce 931, má čtyři metody:

1. float getQuote(String symbol)
2. void getQuoteOneWay(String symbol).
3. int asyncQuote(int delay)
4. float getQuoteTran(String symbol)

```
package soap.server;
import java.lang.Thread;
import java.io.FileWriter;
public class StockQuoteAxis {
    public float getQuote(String symbol) throws Exception {
        return ((float) 55.25);
    }
    public void getQuoteOneWay(String symbol) throws Exception {
        try {
            // Write the results for this service to a file
            FileWriter f = new FileWriter("getQuoteOneWay.txt", true);
            f.write("One way service result via proxy is: 44.44\n");
            f.close();
        } catch (Exception ee) {
            System.out.println("Error writing result file in getQuoteOneWay");
            ee.printStackTrace();
        }
    }
    public int asyncQuote(int delay) {
        try {
            Thread.sleep(delay);
        } catch (Exception e) {
            System.out.println("Exception in asyncQuote during sleep");
        }
        return delay;
    }

    public float getQuoteTran(String symbol) throws Exception {
        if (symbol.equalsIgnoreCase("ROLLBACK")) {
            System.out.println("Rollback was requested,
                exiting from service by calling System.exit().");
            System.exit(0);
        }
        return ((float) 55.25);
    }
}
```

Obrázek 170. Osa StockQuote

Jak pokračovat dále

Implementujte službu pomocí produktu WebSphere MQ Transport pro protokol SOAP, nikoli pomocí protokolu HTTP pomocí příkazu **amqwdeployWMQService**.

Tento příkaz má volbu `axisDeploy`, která implementuje službu vytvořením deskriptoru implementace Apache Axis 1.4 . Modul listener protokolu SOAP produktu WebSphere MQ spouští službu. Modul listener SOAP se nazývá listener SimpleJavaa je dodáván s přenosem protokolu SOAP produktu WebSphere MQ .

Související úlohy

Vývoj služby .NET 1 nebo 2 pro přenos WebSphere MQ pro SOAP pomocí produktu Microsoft Visual Studio 2008

Vývoj webové služby Quote pro SampleStockpro prostředí .NET 1 nebo .NET 2 pomocí produktu Microsoft Visual Studio 2008

Vývoj webové služby JAX-WS EJB pro W3C SOAP prostřednictvím rozhraní JMS

Webová služba, která je svázána s doporučujícím doporučením W3C pro protokol SOAP prostřednictvím rozhraní JMS, musí být spuštěna v kontejneru EJB aplikačního serveru JEE. Tato úloha je krokem 2 připojení klienta webové služby Axis2 a webové služby implementované na serveru WebSphere Application Server pomocí protokolu W3C protokolu SOAP prostřednictvím rozhraní JMS.

Vývoj služby .NET 1 nebo 2 pro přenos WebSphere MQ pro SOAP pomocí produktu Microsoft Visual Studio 2008

Vývoj webové služby Quote pro SampleStockpro prostředí .NET 1 nebo .NET 2 pomocí produktu Microsoft Visual Studio 2008

Informace o této úloze

Vytvořte službu StockQuote s implementací kódu za použití produktu Visual Studio 2008.

Postup

1. Vytvořte šablonu pro službu a zkontrolujte, zda je spuštěna na protokolu HTTP.
   a) Start Visual Studio 2008 > **Soubor** > **Nový** > **Projekt ...**. Vyberte volbu **C#** Typ projektu, **, NET Framework 2**a **ASP.NET Aplikace webové služby**. Zadejte **Název:** a **Název řešení:** `StockQuoteDotNet` > **OK**
   b) Pravým tlačítkem myši klepněte na položku **Service1.asmx** v části **Průzkumník řešení** > **Přejmenovat** > `StockQuote.asmx`.
   c) Změňte fragment kódu `public class Service1` na `public class StockQuote`.
   d) Pravým tlačítkem myši klepněte na položku **StockQuote.asmx** v **Průzkumníku řešení** > **Otevřít pomocí ...** > **Editor XML**. Změnit `Class="StockQuoteDotNet.Service1"` na `Class="StockQuoteDotNet.StockQuote"`
   e) Změňte fragment kódu `[WebService(Namespace = "http://tempuri.org/")]` na `[WebService(Namespace = "http://stock.samples/")]`.
   f) Odeberte řádek kódu `[ToolboxItem(false)]`.
   g) Zkontrolujte vše, co je zatím správné: **Debug** > **Start Debugging (F5)**. Ověřte výstup v Průzkumníku.
2. Přidejte metody z ukázky `SQDNNonInline.asmx.cs`a otestujte službu na HTTP.
   a) Otevřete produkt *MQ_INSTALLATION_PATH*`\tools\soap\samples\dotnet\SQDNNonInline.asmx.cs` a nahraďte metodu `HelloWorld` čtyřmi metodami `Quote` ; viz Obrázek 171 na stránce 934. *MQ_INSTALLATION_PATH* Představuje adresář, kde je nainstalován produkt WebSphere MQ .
   b) **Sestavit** > **Znovu sestavit** řešení > Klepněte pravým tlačítkem myši na jeden z **Podproces** v chybě > **Vyřešit** > Použití **System.Threading**.
   c) Stisknutím klávesy F5 spustíte ladění programu.

Služba není souhlasná s profilem WS-I Basic Profile v1.1. Máte volbu buď změnit anotaci WebMethod z `[SoapRpcMethod]` na `[SoapDocumentMethod]`, nebo odebrat anotaci `[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]`.

d) Stisknutím klávesy F5 ověřte implementaci pomocí protokolu HTTP.

3. Generovat WSDL, klienty a spustit službu pomocí přenosu WebSphere MQ pro SOAP.
   a) Otevřete příkazové okno ve stromu adresáře projektu, kde je uložen `StockQuote.asmx` .
   b) (Volitelné) Chcete-li generovat artefakty, použijte amqwdeployWMQService . Správce front musí být spuštěn:

```
amqswdeployWMQService -f StockQuote.asmx
-u "jms:/queue?initialContextFactory=com.ibm.mq.jms.Nojndi
&connectionFactory=()
&destination=REQUESTDOTNET@QM1
&targetService=StockQuote.asmx"
StockQuote.asmx StockQuote.wsdl
```

   Všechny artefakty se vytvoří ve stromu adresáře `./generated` .

   c) (Volitelné) Generujte pouze WSDL pro volání služby pomocí přenosu WebSphere MQ pro SOAP.

```
amqswsdl -u "jms:/queue?initialContextFactory=com.ibm.mq.jms.Nojndi
&connectionFactory=()
&destination=REQUESTDOTNET@QM1
&targetService=StockQuote.asmx"
StockQuote.asmx StockQuote.wsdl
```

   a) Spusťte modul listener .NET. Buď použijte `.\generated\server\startWMQNListener.cmd` , nebo zadejte příkaz:

```
amqSOAPNETListener -u "jms:/queue?initialContextFactory=com.ibm.mq.jms.Nojndi
&connectionFactory=()
&destination=REQUESTDOTNET@QM1
&targetService=StockQuote.asmx"
```

4. Otestujte službu pomocí klienta generovaného ze souboru WSDL nebo pomocí klientů vygenerovaných produktem **amqwdeployWMQService**.

Ukázkový kód

Vzorová webová služba .NET, `StockQuoteDotNet`, je instalována v *MQ_INSTALLATION_PATH*`\tools\soap\samples\dotnet`. *MQ_INSTALLATION_PATH* je adresář, kde je nainstalován produkt WebSphere MQ . Vazba webových služeb publikovaných ukázek se mírně liší od vazby použité v rámci úlohy. Úloha používá výchozí hodnoty použité v produktu Visual Studio 2008.

K dispozici jsou dva příklady webových služeb .NET Framework 1 a .NET Framework 2. `StockQuoteDotNet.asmx`, je vložená služba. `SQDNNoninline.asmx`, je kódová stránka webová služba implementovaná serverem `SQDNNoninline.asmx.cs`.

`StockQuoteDotNet` má čtyři metody:

1. `float getQuote(String symbol)`
2. `void getQuoteOneWay(String symbol).`
3. `int asyncQuote(int delay)`
4. `float getQuoteDOC(String symbol)`

```
<%@ WebService Language="C#" Class="StockQuoteDotNet" %>
using System;
using System.Web.Services;
using System.Web.Services.Protocols;
using System.Web.Services.Description;
using System.Threading;
[WebService (Namespace="http://stock.samples")]
public class StockQuoteDotNet {
     [WebMethod] [ SoapRpcMethod(OneWay=true) ]
     public void getQuoteOneWay(String symbol) {
          if (symbol.ToUpper().Equals("DELAY")) Thread.Sleep(5000);
          System.Console.WriteLine("getQuoteOneWay was invoked.");
     }
     [WebMethod] [SoapRpcMethod]
     public float getQuote(String symbol) {
          if (symbol.ToUpper().Equals("DELAY")) Thread.Sleep(10000);
          return 88.88F;
     }
     [WebMethod] [SoapRpcMethod]
     public int asyncQuote(int delay) {
          Thread.Sleep(delay);
          return delay;
     }
     [WebMethod]
     public float getQuoteDOC(String symbol) {
          return 77.77F;
     }
}
```

Obrázek 171. Vložená služba: `StockQuoteDotNet.asmx`

```
<%@ WebService Language="C#" Codebehind="SQDNNonInline.asmx.cs" Class="SQDNNonInline" %>
```

Obrázek 172. Kód-za: Návrh `SQDNNonInline.asmx`

```
using System;
using System.Web.Services;
using System.Web.Services.Protocols;
using System.Web.Services.Description;
using System.Threading;

[WebService(Namespace = "http://stock.samples")]
public class SQDNNonInline : System.Web.Services.Protocols.SoapHttpClientProtocol
{

    [WebMethod]
    [SoapRpcMethod(OneWay = true)]
    public void getNonInlineQuoteOneWay(String symbol)
    {
        if (symbol.ToUpper().Equals("DELAY")) Thread.Sleep(5000);
        System.Console.WriteLine("getNonInlineQuoteOneWay was invoked.");
    }

    [WebMethod]
    [SoapRpcMethod]
    public float getNonInlineQuote(String symbol)
    {
        if (symbol.ToUpper().Equals("DELAY")) Thread.Sleep(10000);
        return 88.88F;
    }

    [WebMethod]
    [SoapRpcMethod]
    public int asyncNonInlineQuote(int delay)
    {
        Thread.Sleep(delay);
        return delay;
    }
    [WebMethod]
    public float getNonInlineQuoteDOC(String symbol)
    {
        return 77.77F;
    }
}
```

Obrázek 173. Kód-za: Implementace: SQDNNonInline.asmx.cs

Související úlohy

Vývoj služby JAX-RPC pro přenos WebSphere MQ pro SOAP pomocí prostředí Eclipse
Vyvinout webovou službu Axis 1.4 , která má být spuštěna s použitím produktu WebSphere MQ jako poskytovatele služeb. Použijte normální prostředí pro vývoj webových služeb k vytvoření služby pro implementaci na Axis 1.4.

Vývoj webové služby JAX-WS EJB pro W3C SOAP prostřednictvím rozhraní JMS
Webová služba, která je svázána s doporučujícím doporučením W3C pro protokol SOAP prostřednictvím rozhraní JMS, musí být spuštěna v kontejneru EJB aplikačního serveru JEE. Tato úloha je krokem 2 připojení klienta webové služby Axis2 a webové služby implementované na serveru WebSphere Application Server pomocí protokolu W3C protokolu SOAP prostřednictvím rozhraní JMS.

Vývoj webové služby JAX-WS EJB pro W3C SOAP prostřednictvím rozhraní JMS

Webová služba, která je svázána s doporučujícím doporučením W3C pro protokol SOAP prostřednictvím rozhraní JMS, musí být spuštěna v kontejneru EJB aplikačního serveru JEE. Tato úloha je krokem 2 připojení klienta webové služby Axis2 a webové služby implementované na serveru WebSphere Application Server pomocí protokolu W3C protokolu SOAP prostřednictvím rozhraní JMS.

Než začnete

Pomocí produktu Rational Application Developer vytvořte webovou službu EJB. Průvodce webovými službami v produktu Rational Application Developer má možnost vytvořit webovou službu pomocí doporučeného doporučení W3C pro vazbu SOAP prostřednictvím JMS. Produkt Rational Application Developer 7.54 je povinný. Cvičení používá produkt Rational Application Developer zahrnutý v produktu Rational Software Architect pro produkt WebSphere Software v7.5.5.1,

Objekt EJB je implementován na server WebSphere Application Server z produktu Rational Application Developer jako součást této úlohy. Musíte dokončit “Konfigurace serveru WebSphere Application Server pro použití W3C SOAP přes JMS” na stránce 973

Chcete-li vytvořit WSDL skutečně použité v úloze, musíte nejprve dokončit úlohu, “Vývoj služby JAX-RPC pro přenos WebSphere MQ pro SOAP pomocí prostředí Eclipse” na stránce 928. Poté můžete buď importovat WSDL z dynamického webového projektu v pracovním prostoru Galileo platformy Eclipse , nebo ze spuštěné webové služby HTTP implementované do WASCE.

Server WebSphere Application Server může být stále spuštěn jako výsledek provedení “Konfigurace prostředků serveru WebSphere Application Server” na stránce 975. Pokud tomu tak není, můžete jej spustit ze zobrazení Servery v RAD.

Informace o této úloze

In this task, you redeploy the `StockQuoteAxis` service from running as a JAX-RPC Axis service run by the **`SimpleJavaListener`** using WebSphere MQ transport for SOAP, to being a JAX-WS service running in WebSphere Application server using the W3C SOAP over JMS protocol.

Pro migraci služby z produktu **`SimpleJavaListener`** na server WebSphere Application Serverjsou k dispozici dvě části:

1. Generujte rozhraní webové služby ze souboru WSDL pro danou službu pomocí průvodce webovou službou Top-down EJB v produktu Rational Application Developer.
2. Implementace služby naimportováním ukázky protokolu SOAP produktu WebSphere MQ `StockQuoteAxis.java`.

Alternativním přístupem by bylo generování služby zdola nahoru, ze serveru `StockQuoteAxis.java`. Chcete-li však mít jistotu, že rozhraní migrované služby je identické, je přístup shora dolů lepší, protože používá stejný WSDL.

Webová služba je vyvíjeny pro kontejner EJB, nikoli pro webový kontejner, protože podpora platformy JMS je součástí kontejneru EJB.

Postup

1. Spusťte produkt Rational Application Developera ověřte, že je server WebSphere Application Server spuštěný.
   a) Spusťte produkt Rational Application Developer v novém pracovním prostoru.
   b) Otevřete perspektivu Java EE .
   c) Otevřete kartu **Servery** a zkontrolujte, zda je spuštěný server WebSphere Application Server.
      - Pokud v pohledu neexistuje žádný server WebSphere Application Server v7.0 , klepněte pravým tlačítkem myši v zobrazení > **Nový** > **Server**. Postupujte podle voleb v průvodci a vytvořte instanci serveru WebSphere Application Server v7.0 .
      - Je-li server přítomen, ale není spuštěn, spusťte jej klepnutím na šipku se šipkou.
      - Chcete-li ověřit vlastnosti a získat rychlý přístup k protokolům serveru, klepněte pravým tlačítkem myši na **WebSphere Application Server v7.0 na lokálním hostiteli** > **Vlastnosti** > **WebSphere Application Server**.
      - Chcete-li spravovat server, buď použijte externí prohlížeč a otevřete adresu URL, `http://localhost:9061/ibm/console/unsecureLogon.jsp`, nebo klepněte pravým tlačítkem myši na **WebSphere Application Server v7.0 na lokálním hostiteli** > **Spustit administrativní konzolu**.
      - Výchozí nastavení je publikovat automaticky. Mnoho lidí preferuje ruční nasazení aktualizací na server. Poklepejte na volbu **WebSphere Application Server v7.0 na lokálním hostiteli**a rozbalte prvek **Publikování** se šipkou v okně**Přehled** . Klepněte na tlačítko **Nikdy nepublikovat automaticky**.

- Další výchozí hodnotou, kterou byste mohli chtít změnit, je zrušit zaškrtnutí políčka **Ukončit práci serveru při vypnutí pracovní plochy** v okně **Přehled** .

2. Vytvořit projekty JEE

   Musíte vytvořit projekt Enterprise Application Project (EAR) a projekt EJB (Enterprise Java Bean).

   a) **Soubor** > **Nový** > **Projekt podnikové aplikace**. Pojmenujte projekt `W3CJMSEAR` > **Dokončit**.

      Výchozí nastavení musí identifikovat server WebSphere Application Server v7.0 jako cílové běhové prostředí a verzi EAR 5.0. Musí být vybrána výchozí konfigurace.

   b) **Soubor** > **Nový** > **Projekt EJB**. Pojmenujte projekt `W3CJMSEJB`. Vyberte volbu `W3CEARJMS` jako **Název projektu EAR** > **Další**.

      Výchozí verze modulu EJB je 3.0 a znovu se použije výchozí konfigurace.

   c) Zrušte zaškrtnutí políčka **Vytvořit modul JAR klienta EJB** > **Dokončit**.

3. Vygenerujte a implementujte webovou službu EJB ze souboru WSDL produktu `StockQuoteAxis` .

   a) **Spustit** > **Spustit průzkumník webových služeb**.

   b) Vyberte stránku WSDL pomocí ikon v okně**Průzkumník webových služeb** > klepněte na položku **WSDL main** v modulu Navigator.

   c) V okně **Akce** zadejte nebo přejděte na adresu URL WSDL pro produkt `StockQuoteAxis.wsdl`.

      Pokud jste WASCE spustili s produktem `StockQuoteAxis` implementovaným jako služba HTTP, adresa URL je:

      ```
      http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis?wsdl
      ```

      Máte-li v systému souborů WSDL, adresa URL může být:

      ```
      File:\Dirpath\StockQuoteAxis\WebContent\wsdl\StockQuoteAxis.wsdl
      ```

   d) Klepněte na řádek obsahující importovanou adresu URL ve stromu Navigator .

      Jedná se o řádek bezprostředně následující po **WSDL-hlavní**, pokud se jedná o první WSDL, který jste importovali do průzkumníku webových služeb.

   e) V okně **Akce** klepněte na volbu **Spustit průvodce webovou službou** > **Skepleton webové služby** > **Spustit** .

   f) V průvodci webovou službou vyberte volbu **Nejvyšší webová služba EJB** .

      Vyberte nebo ověřte konfiguraci pomocí informací z volby Tabulka 141 na stránce 937 Zkontrolovat **přepsat soubory bez varování** > **Další**.

      Tabulka 141. Konfigurace webové služby EJB shora dolů

Pole	**Hodnota**
Server	`WebSphere Application Server v7.0`
Doba běhu webové služby	`IBM WebSphere JAX-WS`
projekt služby	`W3CJMSEJB`
Projekt EAR služby	`W3CJMSEAR`
Konfigurace:	`No client generation`

   g) Na stránce pod titulkem **Zadat volby pro vytvoření webové služby WebSphere JAX-WS-EJB shora dolů**zaškrtněte políčko **Přepnout na vazbu JMS**. Zkontrolujte také volbu **Povolit styl obalu**, **Kopírovat WSDL do projektu** a **Generovat deskriptor implementace webové služby** > **Další**.

   h) Na stránce s názvem **Konfigurace vazeb JMSWebSphere JAX-WS**zaškrtněte políčko **Použít protokol interoperability SOAP/JMS** a zadejte hodnoty z produktu Tabulka 142 na stránce 938a ponechejte ostatní pole prázdná > **Další**.

Tabulka 142. WebSphere Konfigurace vazeb JMS platformy JMS	
Pole	**Hodnota**
Cíl služby JMS	queue
Název rozhraní JNDI cíle:	requestaxis
Továrna připojení rozhraní JMS	qm1
Odpovědět na jméno	W3CJMSEAR
Konfigurace:	replyaxis

a) Na stránce s názvem **Konfigurace projektu směrovače WebSphere JAX-WS**zadejte qm1as do pole **Název JNDI ActivationSpec** > **Další**.

RAD trvá asi 30 sekund na minutu pro generování a nasazení projektu.

b) Ignorujte volby na stránce **Publikace webové služby** > **Dokončit**.

4. Zkontrolujte vygenerovaný kód WSDL.

Požádali jste o vygenerování a uložení kódu WSDL specifického pro službu v projektu.

a) V navigátoru podnikového průzkumníka otevřete složku **W3CJMSEJB** > **ejbmodule** > **META-INF** > **wsdl**. Poklepejte na tlačítko StockQuoteAxis.wsdl a otevřete jej v editoru WSDL.

Zkontrolujte vazbu; zobrazí se adresa URL služby JMS:

```
jms:jndi:requestaxis?jndiConnectionFactoryName=qm1&targetService=StockQuoteAxis
```

5. Volitelný krok: Svázání EJB s protokolem SOAP prostřednictvím protokolu HTTP pomocí rozhraní JAX-WS.

Poskytnutí dvou vazeb k sadě EJB poskytuje klientům volbu vazeb SOAP pro volání webové služby. Poskytuje klientům také prostředky k dotazování na webový server za účelem získání jeho kódu WSDL pomocí protokolu HTTP.

Kroky pro vytvoření vazby EJB na SOAP přes protokol HTTP nejsou zahrnuty jako součást úlohy.

6. Implementovat a znovu implementovat produkt StockQuoteAxis s použitím ukázky StockQuoteAxis.java

a) V navigátoru podnikového průzkumníka otevřete složku **W3CJMSEJB** > **Služby** Poklepejte na ikonu StockQuoteAxisService a otevřete implementační třídu v editoru Java.

b) Otevřete ukázkový program StockQuoteAxis.java ve složce *WebSphere MQ Installation directory*\tools\soap\samples\java\server > Vyberte všechny metody, ale ne název třídy > **Kopírovat**.

c) V produktu StockQuoteAxisSoapBindingImpl.java vyberte všechny metody, nikoli však název třídy, a vložte je do metod z produktu StockQuoteAxis.java.

d) Přidejte tiskový příkaz na výstup do konzoly WebSphere Application Server při volání této služby. Změňte metodu getQuote(symbol řetězce):

```
public float getQuote(String symbol) {
    System.out.println("StockQuoteAxisSoapBindingImpl called with symbol: "
                       + symbol);
    return ((float) 55.25);
```

e) Opravte importy: **Zdroj** > **Uspořádat importy** > **Uložit**.

f) Opravte tyto tři chyby v důsledku implementace, která neodpovídá rozhraní.

Chyby jsou způsobeny třemi z metod v části StockQuoteAxis.java výjimek a kódem WSDL pro službu, která neobsahuje žádné zprávy o poruchách. Problém je diagnostikován jako nesoulad mezi podpisy metody a anotacemi webové služby metody.

Buď anotujte metody pomocí znaku @WebFault a znovu vygenerujte WSDL, nebo zachovejte rozhraní beze změny a odeberte výjimky.

Chcete-li zachovat rozhraní stejné, odeberte tři `throws exception` z podpisů metod > **Uložit**.

Jak pokračovat dále

“Implementace na klienta Axis2 pomocí protokolu W3C SOAP prostřednictvím rozhraní JMS” na stránce 984

Související úlohy

Vývoj služby JAX-RPC pro přenos WebSphere MQ pro SOAP pomocí prostředí Eclipse
Vyvinout webovou službu Axis 1.4 , která má být spuštěna s použitím produktu WebSphere MQ jako poskytovatele služeb. Použijte normální prostředí pro vývoj webových služeb k vytvoření služby pro implementaci na Axis 1.4.

Vývoj služby .NET 1 nebo 2 pro přenos WebSphere MQ pro SOAP pomocí produktu Microsoft Visual Studio 2008
Vývoj webové služby Quote pro SampleStockpro prostředí .NET 1 nebo .NET 2 pomocí produktu Microsoft Visual Studio 2008

Vývoj klientů webových služeb WebSphere MQ pro produkt WebSphere MQ pro protokol SOAP

Použijte normální vývojové prostředí pro vývoj klientů webových služeb pro použití s přenosem WebSphere MQ pro protokol SOAP.

Než začnete

Vytvořte službu. Můžete použít jeden z příkladů v produktu “Vývoj webových služeb pro přenos WebSphere MQ pro SOAP” na stránce 927.

Vyberte volby, jak vyvíjet, implementovat a používat klienty, a kde získat WSDL pro generování klienta.

Rozhodněte se pro váš přístup k vývoji klientů a služeb pro přenos WebSphere MQ pro SOAP.

Existují dva přístupy.

1. Použijte standardní vývojové nástroje, vytvořte službu HTTP a klienta a potom použijte adresu URL pro přenos WebSphere MQ pro SOAP.
2. Použijte nástroje a ukázky dodávané spolu s přenosem WebSphere MQ pro protokol SOAP.

Pokud použijete trasu HTTP, můžete službu spustit na serveru HTTP a spustit ji také pomocí přenosu WebSphere MQ pro SOAP. Chcete-li ji spustit pomocí přenosu produktu WebSphere MQ pro protokol SOAP, nakonfigurujte příslušný modul listener produktu WebSphere MQ pro protokol SOAP a nastavte cesty a deskriptory implementace pro spuštění služby. Konfigurace produktu WebSphere MQ pro protokol SOAP vám poskytne konfiguraci pro vás. Volitelně můžete prostředí nakonfigurovat tak, aby spouštěly listenery.

Nástroje dodávané s přenosem WebSphere MQ pro SOAP jsou užitečné při zahájení a učení se způsobu implementace přenosu. Pro provozní práci je výhodné používat standardní nástroje a implementovat stejnou službu přístupnou pro různé transporty SOAP.

Rozhodněte se, jaký typ klienta se má vyvíjet

Je třeba rozhodnout, jaký typ klienta webové služby se má vyvíjet. Volba závisí na tom, zda znáte rozhraní služby a adresu služby.

Je-li rozhraní známo, použijte nástroje Axis nebo .NET ke generování tříd klienta proxy z rozhraní služby. Třídy klientů proxy zjednodušují zápis klienta pro volání služby. Je-li umístění služby známé při vývoji klienta, použijte statické rozhraní proxy. Pokud se změní umístění služby, například pokud je služba znovu implementována na produkční server, použijte dynamické rozhraní serveru proxy.

Není-li rozhraní služby známo v době, kdy vyvíjíte klienta, na ose Axis, můžete vytvořit klienta DII (Dynamic Invocation Interface) pro Axis 1.4. Klient DII používá generické rozhraní k volání libovolné služby. Chcete-li předat parametry konkrétní službě správně, musíte sestavit specifické rozhraní služby programově. Sestavte rozhraní programově v klientovi nebo načítáním WSDL pro

danou službu do klienta. V prostředí Axis2můžete vytvořit klienta Dispatch. Klient Dispatch používá model dokumentu k popisu požadavku klienta, zatímco klient DII používá model volání. Obě pracují na sestavení požadavku dynamicky.

Získat WSDL pro službu

S výjimkou případu, kdy se rozhraní služby sestavuje programově, musíte nejprve získat WSDL služby, abyste vytvořili klienta webové služby. Služba WSDL je možné získat ze tří různých zdrojů:

1. Přímo z implementace webové služby pomocí nástroje jako např. **java2wsdl** (Axis) nebo **disco** (.NET).
2. Dotazem na webovou službu pomocí adresy URL: *Web service http url*?wsdl.
3. Ze souboru, buď na systému souborů, nebo z registru, jako je UDDI nebo WebSphere Service Registry and Repository.

Poznámka: Není-li služba přístupná pomocí protokolu HTTP, pak dotaz WSDL nefunguje. Samotná služba může být k dispozici pouze prostřednictvím transportu produktu WebSphere MQ pro protokol SOAP.

Kód WSDL generovaný produktem **amqwdeployWMQService** není stejný jako soubor WSDL generovaný pomocí produktu **java2wsdl** nebo **disco**. Vygenerovaný kód WSDL se také liší od všech WSDL, které jste mohli začít s vytvořením služby "Top Down". Na ose Axis mapuje deskriptor implementace produktu `server-config.wsdd` zprávu SOAP vytvořenou klientem na operaci a službu. Produkt **amqwdeployWMQService** generuje jiný deskriptor implementace z platformy Eclipse.

WSDL, který používáte k sestavení klientů, závisí na způsobu implementace služby:

Implementováno pomocí amqwdeployWMQService
Použijte kód WSDL generovaný produktem **amqwdeployWMQService** . Zadejte příznak -w a vyberte volbu `rpcLiteral` WSDL. Z důvodů kompatibility můžete vybrat volbu `rpcEncoded` WSDL. `rpcEncoded` WSDL funguje pouze s klienty .NET a Axis 1.4 .

Ruční implementace pomocí produktu SimpleJavaListener
Použijte jeden z následujících souborů WSDL:

1. Kód WSDL použitý k definování služby nebo uložení v úložišti.
2. Jazyk WSDL generovaný ze služby produktem **java2wsdl** .
3. Dotazovaný kód WSDL s použitím adresy URL *Web service http url* ?wsdl, pokud je k dispozici na serveru HTTP. Chcete-li importovat definici služby přímo do platformy Eclipse, můžete spustit nástroj, jako např. průzkumník webových služeb.

Možná budete muset změnit identifikátor URI pro službu. Změňte ji z adresy služby HTTP na identifikátor URI pro přenos WebSphere MQ pro SOAP.

Ruční implementace pomocí produktu amqSOAPNETListener.
Použijte jeden z následujících souborů WSDL:

1. Kód WSDL použitý k definování služby nebo uložení v úložišti.
2. Soubor WSDL získaný ze třídy služeb .NET (.asmx). pomocí produktu **disco**.
3. Dotazovaný kód WSDL s použitím adresy URL *Web service http url* ?wsdl, pokud je k dispozici. Chcete-li importovat definici služby přímo do platformy Eclipse, můžete spustit nástroj, jako např. průzkumník webových služeb.
4. Kód WSDL získaný spuštěním produktu **amqswsdl** proti třídě služeb .NET (.asmx).

Možná budete muset změnit identifikátor URI pro službu. Změňte ji z adresy služby HTTP na identifikátor URI pro přenos WebSphere MQ pro SOAP.

Nasazeno na Windows Communication Foundation
Službu WSDL získejte pomocí adresy URL *Web service http url*?wsdl. Služba musí být definována pomocí konfigurace chování `serviceMetaData` jako součást definice služby.

Implementace na jinou platformu serveru.
Postupujte podle pokynů poskytnutých spolu s platformou o tom, jak získat správný kód WSDL služby.

Informace o této úloze

Vyvíjejte klienty pomocí standardních vývojových nástrojů. Následující úlohy znázorňují, jak sestavit klienty pro prostředí .NET 1 a 2, Axis 1.4 (JAX-RPC) a Axis2 (JAX-WS). Pro Windows Communication Foundation si prohlédněte odkazy na související úlohy.

Vývoj klienta JAX-RPC pro transport WebSphere pro protokol SOAP pomocí prostředí Eclipse

Vývoj klienta webové služby Axis 1.4 pro použití přenosu protokolu SOAP v produktu WebSphere MQ .

Než začnete

Musíte mít k dispozici službu. Pokud sledujete úlohu jako praktické cvičení, použijte pracovní prostor a službu, které jste vytvořili v úloze “Vývoj služby JAX-RPC pro přenos WebSphere MQ pro SOAP pomocí prostředí Eclipse” na stránce 928. Musíte mít spuštěný aplikační server na platformě Eclipse, která podporuje webové služby Axis 1.4 . V této úloze používáme volně dostupné produkty WebSphere Application Server Community Edition verze 2.1.4. Je nakonfigurován jako součást úlohy “Vývoj služby JAX-RPC pro přenos WebSphere MQ pro SOAP pomocí prostředí Eclipse” na stránce 928. Můžete také použít Tomcat 6, což je menší aplikační server s otevřeným zdrojovým kódem.

Informace o této úloze

Tato úloha ukazuje vývoj tří typů klientů pro ukázkovou službu Axis StockQuotepomocí platformy Eclipse spuštěnou v systému Windows. Klienti jsou statický a dynamický klient vyvinutý pomocí klienta proxy klienta a klienta DII.

Jsou ilustrovány dva alternativní přístupy ke generování proxy klienta z WSDL:

1. Generování proxy klienta pomocí produktu **amqwdeployWMQService** .
2. Import WSDL do platformy Eclipsea použití průvodce webovou službou pro generování proxy klienta.

Postup

1. Spusťte vývojáře Eclipse IDE pro vývojáře Java EE .
2. Vytvořte projekt Java s názvem `StockQuoteAxisClient`:
   a) Přepněte na perspektivu Java > **Soubor** > **Nový** > **Projekt Java**. V poli **`Project name`** na typu **Vytvořit stránku projektu Java** `StockQuoteAxisEclipseClient`. Ujistěte se, že prováděcí prostředí je buď **J2SE1-1.4** , nebo **J2SE-1.5** > **Další** .
   b) Na stránce **Nastavení prostředí Java** vyberte kartu **Knihovny** > **Přidat externí soubory JAR ...**
   c) Přejděte na *MQ_INSTALLATION_PATH*`/java/lib` a vyberte všechny soubory `.jar` > **Otevřít**.
   MQ_INSTALLATION_PATH je adresář, kde je nainstalován produkt WebSphere MQ .
   d) Přejděte na *MQ_INSTALLATION_PATH*`/java/lib/soap` a vyberte všechny soubory `.jar` > **Otevřít**. Musíte mít nainstalován produkt `axis.jar` z instalačního média produktu WebSphere MQ do tohoto adresáře.
   MQ_INSTALLATION_PATH je adresář, kde je nainstalován produkt WebSphere MQ .
   e) Karta **Knihovna** nyní odkazuje na všechny soubory `.jar` potřebné k sestavení klienta > **Dokončit**.
3. Postupujte podle jednoho z těchto dvou přístupů k vytvoření serverů proxy v prostředí Eclipse pro ukázkovou webovou službu Axis StockQuote:
   - Generujte proxy klienta pomocí produktu **amqwdeployWMQService** .
     a. Vytvořte správce front. Pro úlohu create `QM1` jako výchozího správce front.

b. Vytvořte pracovní adresář, `samples`. Zkopírujte ukázkový program `StockQuoteAxis.java` do produktu `samples/soap/server`.

c. Upravte `amqwsetcp.cmd` v produktu *MQ_INSTALLATION_PATH*`/bin` tak, aby obsahoval aktuální adresář v cestě ke třídě. *MQ_INSTALLATION_PATH* je adresář, kde je nainstalován produkt WebSphere MQ .

d. Otevřete příkazové okno v produktu `samples` a spusťte upravený příkaz **amqwsetcp** .

e. Spuštěním příkazu vytvořte WSDL pro službu Axis StockQuote.

```
amqwdeployWMQService -f soap/server/StockQuoteAxis.java -c genAxisWsdl
-u "jms:/queue?destination=REQUESTAXIS
&initialContextFactory=com.ibm.mq.jms.Nojndi
&connectionFactory=(connectQueueManager(QM1)binding(auto))"
```

Zapamatujte si: Používáte-li příkazy jazyka Java, používejte raději "`/`"než "`.`" nebo "`\`" .

Tip: Místo importu vygenerovaných serverů proxy do platformy Eclipsemůžete importovat generovaný WSDL z produktu `.samples/generated`. Výsledné servery proxy se liší dvěma způsoby:

i) Názvy balíků se liší-které můžete refaktorovat.

ii) Servery proxy generované produktem Eclipse obsahují přídavnou třídu pomocníka `StockQuoteAxisProxy.java`

f. Vytvořte proxy klienta pro službu Axis StockQuotetak, že spustíte příkaz:

```
amqwdeployWMQService -f soap/server/StockQuoteAxis.java -c genProxiestoAxis
-u "jms:/queue?destination=REQUESTAXIS
&initialContextFactory=com.ibm.mq.jms.Nojndi
&connectionFactory=(connectQueueManager(QM1)binding(auto))"
```

g. Naimportujte proxy klienta do StockQuoteAxisClient:

i) Pravým tlačítkem myši klepněte na položku **StockQuoteAxisClient\src** > Vyberte **Systém souborů** > **Další** > **Procházet ...** > najděte složku `.\samples\generated\client\remote\soap\server` > **OK**.

ii) Zkontrolujte **server** na stránce**Import** > **Dokončit**.

h. Refaktorujte název balíku na `soap.server`.

i) Klepněte pravým tlačítkem myši na balík obsahující proxy klienta > **Refaktorovat** > **Přejmenovat**. Zadejte **New name**: `soap.server` > ponechejte vybrané předvolby pro ostatní volby > **OK** . Všechny chyby jsou opraveny.

- Generujte proxy klienta pomocí Eclipse.

  Máte volbu způsobů, jak získat kód WSDL pro službu. V tomto příkladu byla služba implementována na server WebSphere Application Server Community Edition a vy jste získali soubor WSDL z webového serveru. Implementace je popsána v úloze “Vývoj služby JAX-RPC pro přenos WebSphere MQ pro SOAP pomocí prostředí Eclipse” na stránce 928,

  a. V prostředí Eclipsepřepněte na webovou perspektivu a zkontrolujte, zda je spuštěn server WebSphere Application Server Community Edition v2.1 Server, a že je implementována a synchronizována položka StockQuoteAxis.

  b. Naimportujte WSDL do průzkumníku webových služeb:

  i) Klepněte na ikonu **Průzkumník webových služeb** na řádku s akcemi nebo klepněte na volbu **Spustit** > **Spustit průzkumník webových služeb**.

  ii) Klepněte na ikonu stránky WSDL v průzkumníku webových služeb a přepněte se na stránku WSDL.

  iii) Klepněte na volbu **Hlavní soubor WSDL** v okně Navigator v průzkumníku webových služeb.

iv) Zadejte adresu URL webové služby a za ní ?WSDL . Adresa URL pro osu StockQuoteimplementovaná v úloze “Vývoj služby JAX-RPC pro přenos WebSphere MQ pro SOAP pomocí prostředí Eclipse” na stránce 928je:

```
http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis?wsdl
```

c. Generovat proxy klienta:

i) V navigátoru průzkumníku webových služeb klepněte na volbu **http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis?wsdl** .

ii) V okně **Akce** klepněte na volbu **Spustit průvodce webovou službou** >, chcete-li vybrat volbu **Klient webové služby** > **Přejít**.

iii) Na první stránce průvodce klepněte na odkaz projektu **Klient** v konfiguraci > Vyberte projekt klienta **StockQuoteAxisClient** na první stránce > **OK**.

Tip: Okno průvodce může ztratit fokus. Musíš se vrátit do soustředění ručně.

iv) Běhové prostředí webové služby musí být Apache Axis pro generování klienta JAX-RPC.

v) Klepněte na tlačítko **Dokončit**.

vi) Změňte statickou adresu URL služby tak, aby ukazovala na přenos WebSphere MQ pro adresu SOAP pro službu Axis StockQuote. Můžete zvolit přeskočení tohoto kroku, dokud jste neotestovali klienta se serverem HTTP.

a) Otevřete `StockQuoteAxisServiceLocator.java` a vyhledejte deklaraci pro `StockQuoteAxis_address`.

b) Změnit adresu URL na

```
"jms:/queue?destination=REQUESTAXIS
&amp;initialContextFactory=com.ibm.mq.jms.Nojndi
&amp;connectionFactory=(connectQueueManager(QM1)binding(auto))"
```

Tip: Platforma Eclipse automaticky transformuje & na `&` a naopak, když kopírujete a vložíte řetězce do kódu `.java` .

d. Vytvořte tři třídy klienta Java, každý s hlavní metodou:

i) Vytvořit balík. Pravým tlačítkem myši klepněte na položku **StockQuoteAxisClient/src** > **Nový balík**. Pojmenujte jej `soap.client` > **Dokončit**.

ii) Vyberte volbu **soap.client** > **New** > **Class**. Pojmenujte třídu `SQAStaticClient` > Zkontrolujte **public static void main (string [] args)** > Dokončit

iii) Zopakujte proceduru pro vytvoření `SQADynamicClient.java` a `SQADIIClient.java`

e. Zapište kód klienta.

Obrázek 177 na stránce 947 prostřednictvím produktu Obrázek 181 na stránce 949 poskytují příklady tří stylů kódu klienta. Příklady používají adresu URL HTTP k testování klienta pomocí služby Axis StockQuoteimplementované na server HTTP. Chcete-li spustit klienty na službě Axis StockQuoteimplementovanou pomocí přenosu WebSphere MQ pro SOAP, změňte adresu URL na:

```
"jms:/queue?destination=REQUESTAXIS
connectionFactory=(connectQueueManager(QM1)binding(auto))
initialContextFactory=com.ibm.mq.jms.Nojndi
targetService=soap.server.StockQuoteAxis.java
replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE"
```

– Produkt Obrázek 177 na stránce 947 a produkt Obrázek 179 na stránce 948 používají server proxy generovaný platformou Eclipse, která má navíc pomocnou třídu StockQuoteAxisproxy, která usnadňuje kódování kódu.

– Obrázek 178 na stránce 948 a Obrázek 180 na stránce 948 používají server proxy generovaný produktem **amqwdeployWMQService** .

– Produkt Obrázek 181 na stránce 949 nepoužívá žádné třídy serveru proxy.

Každý z klientů volá produkt `com.ibm.mq.soap.Register.extension()` , aby se propojí s přenosem protokolu SOAP produktu WebSphere MQ . Rozšíření je registrováno v deskriptoru implementace klienta. Implementace klienta do Axis 1.4 je popsána v tématu “Implementace klienta webové služby do Axis 1.4 pro použití přenosu IBM WebSphere MQ pro SOAP” na stránce 979.

f. Spusťte klienty odesláním požadavku SOAP na osu StockQuote, jejímž hostitelem je server WebSphere Application Server Community Edition nakonfigurovaný v pracovním prostoru.

   i) Zkontrolujte, zda je server spuštěn, je implementována a synchronizována položka StockQuoteAxis.

   ii) Vyberte nebo otevřete klienta, kterého chcete testovat > Klepněte na tlačítko **Spustit** na řádku s akcemi. Případně klepněte na zelenou ikonu Spustit nebo na osm klepnutí na klienta v navigátoru > **Spustit jako** > **Spustit konfigurace ...**. Konfigurujte parametry, které potřebujete ke spuštění klienta.

g. Spusťte klienta pomocí přenosu WebSphere MQ pro SOAP.

   Procedura používá produkt **amqwdeployWMQService** k implementaci služby a pracuje pouze s klientem, který používá WSDL nebo servery proxy sestavené produktem **amqwdeployWMQService** . Chcete-li klienta spustit s použitím původních souborů WSDL nebo serverů proxy sestavených platformou Eclipse, implementujte tuto službu s deskriptorem implementace sestavenou platformou Eclipse. Ruční spuštění produktu **SimpleJavaListener** pomocí názvu vazby portu služby jako názvu `targetServiceName` .

   i) Postupujte podle pokynů v části “Implementace služby na Axis 1.4 k použití pro přenos WebSphere pro protokol SOAP pomocí produktu amqwdeployWMQService” na stránce 968 a implementujte službu do jednoduchého modulu listener Java SOAP produktu WebSphere MQ . Implementace služby pracuje pouze pro klienta s použitím serverů proxy WSDL nebo klientů vytvořených produktem **amqwdeployWMQService**.

   ii) V příkazovém okně spusťte příkaz **amqwclientconfig** , abyste vytvořili soubor deskriptoru implementace klienta `client-deploy.wsdd` .

   iii) Naimportujte produkt `client-deploy.wsdd` do kořenového adresáře projektu Java, který chcete testovat pomocí přenosu WebSphere MQ pro SOAP.

      a) Klepněte pravým tlačítkem myši na projekt Java **StockQuoteAxisEclipseClient** > **Import** > **Systém souborů** > **Další** > **Procházet ...**

      b) Přejděte do adresáře, který obsahuje `client-deploy.wsdd` > **Otevřít** > Vyberte adresář na stránce **Importovat** průvodce > zaškrtněte `client-deploy.wsdd` v pravém podokně.

      c) Ověřte, že **Do složky:** bylo zadáno `StockQuoteAxisEclipseClient` > **Dokončit**.

   iv) Potvrďte, že pracovní adresář pro spuštění aplikace Java v tomto projektu je adresář `StockQuoteAxisEclipseClient` :

      Klepněte pravým tlačítkem myši na projekt Java **StockQuoteAxisEclipseClient** > **Spustit jako ...** > **Spustit konfigurace ...** > Vyberte kartu **(x) = Argumenty** > Ověřte, zda je v pracovním adresáři zaškrtnuto políčko **Výchozí** , a cesta je `StockQuoteAxisEclipseClient` . Případně proveďte jednu z následujících voleb pro výběr jiného umístění nebo souboru obsahujícího konfiguraci klienta:

      – Zaškrtněte volbu **Další:** > zadejte cestu k adresáři podle vaší volby.

      – V okně **Argumenty VM** zadejte `-Daxis.ClientConfigFile=`*`full path to client deployment descriptor file`*

   v) Ujistěte se, že adresa URL je nakonfigurována tak, aby ukazovala na službu implementovanou pomocí přenosu WebSphere MQ pro SOAP. Spusťte klienta podle popisu v kroku ii.

   Tip: Obvykle se můžete setkat s jednou z těchto chyb:

   i) `Exception: No client transport named 'jms' found!` .

   ii) Chyba připojení JMS.

iii) `Exception: The AXIS engine could not find a target service to invoke! targetService is soap.server.StockQuoteAxis.java`

iv) `Exception: java.lang.InstantiationException: soap.server.StockQuoteAxis`

Vysvětlení:

i) `client-config.wsdd` nebyl nalezen, nebo zahrnuje řádek `<transport name="jms" pivot="java:com.ibm.mq.soap.transport.jms.WMQSender"/>` v `client-config.wsdd`.

ii) Je možné, že se problém s cestou sestavení-nezahrnuje soubory .jar v produktu *MQ_INSTALLATION_PATH*`/java/lib` . *MQ_INSTALLATION_PATH* je adresář, kde je nainstalován produkt WebSphere MQ .

iii) Problém implementace služby, buď s `server-config-wsdd` , nebo s parametry předanými do **`SimpleSoapListener`** .

iv) Neshoda mezi deskriptorem implementace a implementací služby.

Máte-li potíže se spuštěním klienta v prostředí Eclipse, zkuste použít příkazové okno:

i) Přepněte do adresáře `StockQuoteAxisEclipseClient\bin` ve stromu adresáře pracovního prostoru.

ii) Spusťte **amqwsetcp** a **amqwclientconfig**

iii) Spusťte příkaz `java soap/client/SQAStaticClient`.

Ukázkové klienty webových služeb JAX-RPC

Ukázkové klienty webové služby Java dodané s produktem WebSphere MQ jsou nainstalovány v produktu *MQ_INSTALLATION_PATH* `\tools\soap\samples\java\clients`. *MQ_INSTALLATION_PATH* je adresář, kde je nainstalován produkt WebSphere MQ .

`SQAxis2Axis.java`
`SQAxis2Axis.java`, Obrázek 174 na stránce 946 , je dynamický klient proxy pro vyvolání služby `StockQuoteAxis` . Adresu URL služby, která se má kompilovat do dynamického serveru proxy, můžete přepsat zadáním adresy URL na příkazovém řádku.

`SQAxis2DotNet.java`
`SQAxis2DotNet.java`, Obrázek 175 na stránce 946, je dynamický klient proxy pro vyvolání služby `StockQuoteDotNet` . Adresu URL služby, která se má kompilovat do dynamického serveru proxy, můžete přepsat zadáním adresy URL na příkazovém řádku.

`WsdlClient.java`
`WsdlClient.java`, Obrázek 176 na stránce 947, je klientem dynamického vyvolání k vyvolání buď služby `StockQuoteDotNet` , nebo `StockQuoteAxis` . Klient vyvolá ve výchozím nastavení službu `StockQuoteAxis` . Přidejte volbu příkazového řádku `-D` , vyvolejte službu `StockQuoteDotNet` a `-w` pro poskytnutí jiného portu v produktu `.\generated\StockQuoteDotNet_Wmq.wsdl` .

```
package soap.clients;
import java.net.URL;
import soap.server.*;
public class SQAxis2Axis {
   public static void main(String[] args) {
      com.ibm.mq.soap.Register.extension();
      try {
         StockQuoteAxisService locator = new StockQuoteAxisServiceLocator();
         StockQuoteAxis service = null;
         if (args.length == 0)
            service = locator.getSoapServerStockQuoteAxis_Wmq();
         else
            service = locator.getSoapServerStockQuoteAxis_Wmq(
                new java.net.URL(args[0]));
         System.out.println("Response: " + service.getQuote("XXX"));
      } catch (Exception e) {
         System.out.println("\n>>> EXCEPTION WHILE RUNNING ProxyClient DEMO <<<\n");
                           e.printStackTrace();
         System.exit(2);
      }
   }
```

Obrázek 174. SQAxis2Axis.java

```
public class SQAxis2DotNet {
public static void main(String[] args) {
   com.ibm.mq.soap.Register.extension();
      try {
         StockQuoteDotNet locator = new StockQuoteDotNetLocator();
         StockQuoteDotNetSoap_PortType service = null;
         if (args.length == 0)
            service = locator.getStockQuoteDotNetSoap();
         else
            service = locator.getStockQuoteDotNetSoap(new java.net.URL(
                  args[0]));
         System.out.println("Response: " + service.getQuoteDOC("XXX"));
      } catch (Exception e) {
         System.out.println("\n>>> EXCEPTION WHILE RUNNING ProxyClient DEMO <<<\n");
                           e.printStackTrace();
         System.exit(2);
      }
   }
}
```

Obrázek 175. SQAxis2DotNet.java

```
package soap.clients;
import com.ibm.mq.soap.*;
import org.apache.axis.utils.Options;
import java.net.URL;
import javax.xml.rpc.Call;
import javax.xml.rpc.Service;
import javax.xml.rpc.ServiceFactory;
import javax.xml.namespace.QName;
public class WsdlClient {
public static void main(String[] args) {
      String wsdlService, wsdlPort, namespace, wsdlSource, wsdlTargetURI, s;
      try {
         Register.extension();
         Options opts = new Options(args);
         if (opts.isFlagSet('D') != 0) {
            wsdlService = "StockQuoteDotNet";
            wsdlPort = "StockQuoteDotNetSoap";
            namespace = "http://stock.samples";
            wsdlSource = "file:generated/StockQuoteDotNet_Wmq.wsdl";
         } else {
            wsdlService = "StockQuoteAxisService";
            wsdlPort = "soap.server.StockQuoteAxis_Wmq";
            namespace = "soap.server.StockQuoteAxis_Wmq";
            wsdlSource = "file:generated/soap.server.StockQuoteAxis_Wmq.wsdl";
         }
         if (null != (s = (opts.isValueSet('w'))))
            wsdlPort = s;
         System.out.println("start WsdlClient demo, wsdl port " + wsdlPort
               + " resolving uri to ...");
         QName servQN = new QName(namespace, wsdlService);
         QName portQN = new QName(namespace, wsdlPort);
         Service service = ServiceFactory.newInstance().createService(
               new URL(wsdlSource), servQN);
         Call call = (Call) service.createCall(portQN, "getQuote");
         wsdlTargetURI = call.getTargetEndpointAddress().toString();
         System.out.println("  '" + wsdlTargetURI + "'");
         Object ret = call.invoke(new Object[] { "XXX" });
         System.out.println("Response: " + ret);
      } catch (Exception e) {
         System.out.println("\n>>> EXCEPTION WHILE RUNNING WsdlClient DEMO <<<\n");
         e.printStackTrace();
         System.exit(2);
      }
   }
}
```

Obrázek 176. WsdlClient.java

Ukázkové klienty použité v této úloze:

```
package soap.client;
import soap.server.StockQuoteAxisProxy;
public class SQAStaticClient {
  public static void main(String[] args) {
    try {
      com.ibm.mq.soap.Register.extension();
      StockQuoteAxisProxy sqa = new StockQuoteAxisProxy();
      System.out.println("Static client synchronous result is:"
          + sqa.getQuote("ibm"));
    } catch (Exception e) {
      System.out.println("Exception: " + e);
    }
  }
}
```

Obrázek 177. Statický klient používající server proxy Eclipse

```
package soap.client;
import soap.server.StockQuoteAxis;
import soap.server.StockQuoteAxisService;
import soap.server.StockQuoteAxisServiceLocator;
public class SQAStaticClient {
  public static void main(String[] args) {
    try {
      com.ibm.mq.soap.Register.extension();
      StockQuoteAxisService locator = new StockQuoteAxisServiceLocator();
      StockQuoteAxis sqa = locator.getSoapServerStockQuoteAxis_Wmq();
      System.out.println("Static client synchronous result is: "
          + sqa.getQuote("ibm"));
    } catch (Exception e) {
      System.out.println("Exception: " + e);
    }
  }
}
```

Obrázek 178. Statický klient používající server proxy amqwdeployWMQService

```
package soap.client;
import soap.server.StockQuoteAxisProxy;
public class SQADynamicClient {
  public static void main(String[] args) {
    try {
      com.ibm.mq.soap.Register.extension();
      StockQuoteAxisProxy sqa = new StockQuoteAxisProxy(
          "http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis");
      System.out.println("Dynamic client synchronous result is: "
          + sqa.getQuote("ibm"));
    } catch (Exception e) {
      System.out.println("Exception: " + e);
    }
  }
}
```

Obrázek 179. Dynamický klient používající server proxy generovaný platformou Eclipse

```
package soap.client;

import java.net.URL;
import soap.server.StockQuoteAxis;
import soap.server.StockQuoteAxisService;
import soap.server.StockQuoteAxisServiceLocator;
public class SQADynamicClient {
  public static void main(String[] args) {
    try {
      com.ibm.mq.soap.Register.extension();
      URL sqaURL = new URL(
          "http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis");
      StockQuoteAxisService locator = new StockQuoteAxisServiceLocator();
      StockQuoteAxis sqa = locator.getSoapServerStockQuoteAxis_Wmq(sqaURL);
      System.out.println("Dynamic client synchronous result is: "
          + sqa.getQuote("ibm"));
    } catch (Exception e) {
      System.out.println("Exception: " + e);
    }
  }
}
```

Obrázek 180. Dynamický klient používající server proxy amqwdeployWMQService

```
package soap.client;
import java.net.URL;
import javax.xml.namespace.QName;
import javax.xml.rpc.Call;
import javax.xml.rpc.Service;
import javax.xml.rpc.ServiceFactory;
public class SQADIIClient {
  public static void main(String[] args) {
    try {
      com.ibm.mq.soap.Register.extension();
      URL wsdl = new URL(
          "http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis?wsdl");
      Service SQAService = (ServiceFactory.newInstance()).createService(wsdl,
          new QName("http://server.soap", "StockQuoteAxisService"));
      Call SQACall = SQAService.createCall(new QName("http://server.soap",
          "StockQuoteAxis"), "getQuote");
      System.out.println("DII client synchronous result is "
          + SQACall.invoke(new Object[] { "ibm" }));
    } catch (Exception e) {
      System.out.println("Exception: " + e);
    }
  }
}
```

Obrázek 181. DII klient (bez proxy)

Související úlohy

Vyvíjení klienta JAX-WS pro přenos WebSphere pro protokol SOAP pomocí prostředí Eclipse
Vytvořte klienta webové služby Axis2 pro použití přenosu protokolu SOAP produktu WebSphere MQ . Ukázkový klient Axis2 poskytovaný s přenosem WebSphere MQ pro protokol SOAP je uveden v seznamu a příkaz **wsimport** použitý ke generování serverů proxy.

Vývoj klienta .NET 1 nebo 2 pro přenos WebSphere pro protokol SOAP pomocí produktu Microsoft Visual Studio 2008
Vytvořte klienta webové služby .NET 1 nebo 2, který má být spuštěn pomocí přenosu WebSphere MQ pro protokol SOAP.

Vyvíjení klienta JAX-WS pro přenos WebSphere pro protokol SOAP pomocí prostředí Eclipse

Vytvořte klienta webové služby Axis2 pro použití přenosu protokolu SOAP produktu WebSphere MQ . Ukázkový klient Axis2 poskytovaný s přenosem WebSphere MQ pro protokol SOAP je uveden v seznamu a příkaz **wsimport** použitý ke generování serverů proxy.

Než začnete

Získejte knihovny Axis2 a nakonfigurujte vývojový a testovací prostředí ke spuštění klienta.

Poznámka: Pojmenování verzí a vydání používaných Axis způsobuje zmatek. Obvykle Axis 1.4 znamená implementaci JAX-RPC, a Axis2 implementaci JAX-WS.

Axis 1.4 je úroveň verze. Když budete hledat Axis 1.4 na internetu, dostanete se na stránku http://ws.apache.org/axis/. Tato stránka obsahuje seznam předchozích verzí Axix (1.2, 1.3) a finální verzi Axis 1.4 z 22. dubna 2006. Existují i starší vydání Axis 1.4, které opravují chyby, ale ty všechny jsou známy jako Axis 1.4. Jedná se o jednu z těchto vydání oprav chyb, které jsou dodávány s produktem WebSphere MQ. Pro osu 1.4použijte verzi produktu `axis.jar` , která je dodávána s produktem WebSphere MQ , a ne tak, aby ji bylo možné získat z adresy http://ws.apache.org/axis/.

Webový server Axis se také odkazuje na Axis 1.1, jenž by se měl týkat všech verzí toho, co je častěji nazýváno Axis 1.4. Axis 1.2 se používá pro vše, co se obvykle nazývá Axis2.

Axis 1.5 není novějším vydáním Axis 1.4, jde o vydání Axis2. Pokud budete hledat Axis 1.5, budete nasměrováni na stránku http://ws.apache.org/axis2/.https://ws.apache.org/axis2/download.cgi Obsahuje seznam verzí produktu Axis2 s popiskem 0.9 až 1.5.1 (včetně matoucí verze 1.4). Verze vydání produktu Axis2 pro použití s přenosem WebSphere MQ pro protokol SOAP je 1.4.1. Stáhněte Axis2 1.4.1 ze stránky http://ws.apache.org/axis2/download/1_4_1/download.cgi.

Můžete zvolit generování proxy pro klienty webových služeb pro přenos WebSphere MQ pro protokol SOAP pomocí produktu **wsimport** nebo nástrojů poskytnutých s prostředím IDE. Eclipse IDE for Java EE Developer 3.5 SR1 používá **wsdl2java**.Produkt **wsimport** je dodáván s jazykem Java 6. Můžete použít jazyk Java 5 ke spuštění proxy klienta generovaných buď s **wsimport** , nebo **wsdl2java**.

Ukázkové klienty webové služby Axis2 dodávané s produktem WebSphere MQ pro protokol SOAP byly vyvinuty pomocí produktu **wsimport**; viz "Ukázka klientů Axis2" na stránce 955.

Následující úloha demonstruje, jak generovat a používat servery proxy vytvořené pomocí průvodce webovými službami, které jsou zabaleny s produktem Eclipse IDE pro vývojáře Java EE . Ukázkové klienty zobrazují způsob použití serverů proxy vytvořených produktem **wsimport**.

Chcete-li použít průvodce webovými službami, je třeba přidat aplikační server, který podporuje prostředí Axis2 , do pracovní plochy. Tyto kroky ukazují, jak nakonfigurovat produkt WASCE tak, aby podporoval Axis2 pomocí pracovní plochy.

1. Nakonfigurujte aplikační server používaný v prostředí Eclipse IDE for Java EE Developers na podporu Axis2. V tomto příkladu nakonfigurujte aplikační server WASCE 2.1.4, který je součástí pracovního prostoru vytvořeného v produktu "Vývoj služby JAX-RPC pro přenos WebSphere MQ pro SOAP pomocí prostředí Eclipse" na stránce 928.
   a. Otevřete předvolby pracovního prostoru a nakonfigurujte server: Otevřete nabídku **Okno** > **Předvolby**.
   b. Zkontrolujte, že nainstalované prostředí JRE je `Java50`: Klepněte na volbu **Instalovaná prostředí JRE**.
   c. Přidejte WASCE jako server: Klepněte na **Server** > **Běhová prostředí** > **Přidat ...** > **IBM** > **WASCE v2.1** > Další. Prostředí JRE musí být Java50 > Přejděte do instalačního adresáře WASCE > **OK** > **Dokončit**. Musíte mít nainstalovaný modul plug-in WASCE pro prostředí Eclipse Java EE IDE for Web Developers.
   d. Přidejte Axis2: Klepněte na **Webové služby** > **Axis2 Předvolby**. Na kartě **Axis2 Běhové prostředí** > **Procházet ...** Otevřete adresář obsahující mnoho souborů JAR produktu `Axis2` > **Použít**.
   e. Přidružte WASCE k Axis2: Klepněte na **Webové služby** > **Server a Běhové prostředí**. Pod **Server** vyberte **IBM WASCE v2.1 Server**a pod **Web service runtime**vyberte **Apache Axis2** > **Použít** > **OK**
   f. Spusťte server: Otevřete webovou perspektivu a otevřete pohled Servery. Klepněte pravým tlačítkem myši na pohled Servery > **Nový** > **Server**. **IBM WASCE v2.1 Server** je vybrán a nakonfigurován > **Dokončit**. Spusťte server.
2. Zkontrolujte, zda jste implementovali službu Axis StockQuotepro produkt WASCE ke spuštění průvodce webovou službou.
3. Chcete-li testovat službu s přenosem WebSphere MQ pro službu SOAP, implementujte službu do transportu produktu WebSphere MQ pro modul listener SOAP pro osu 1.4, viz "Vývoj služby JAX-RPC pro přenos WebSphere MQ pro SOAP pomocí prostředí Eclipse" na stránce 928.

Informace o této úloze

Produkt Eclipse IDE for Java EE Developers používá Java50 a průvodce webovými službami ke generování tříd proxy pro danou službu. Třídy serveru proxy se liší od tříd vytvořených pomocí nástroje **wsimport** poskytovaného s jazykem Java 6. Alternativním přístupem je generování tříd proxy pomocí produktu **wsimport** a import balíků, které vytváří, do prostředí Eclipse Java EE IDE for Web Developers.

Průvodce webovými službami v prostředí Eclipse IDE for Java EE Developer sestavuje klienta webové služby ve webovém projektu. Klienta můžete spustit jako jednoduchou aplikaci Java. Nevyžaduje aplikační server. Můžete také přenést kód do projektu Java a nakonfigurovat cestu sestavení tak, aby obsahovala soubory JAR Axis2 .

Postup

1. Vytvořte webový projekt v novém podnikovém projektu:

a) S ničím vybraným v Průzkumníku projektů > Klepněte pravým tlačítkem myši na prázdný prostor > **Nový** > **Projekt podnikové aplikace** > Název, který je `StockQuoteAxis2EAR` > **Dokončit**. Odpovězte uživateli No do okna, ve kterém můžete otevřít perspektivu Java EE .

Výchozí hodnoty jsou nastaveny na použití WASCE.

b) Klepněte pravým tlačítkem myši na volby `StockQuoteAxis2EAR` > **Nový** > **Dynamický webový projekt**. Pojmenujte projekt `StockQuoteAxis2WebClient` > Zkontrolujte pole členství EAR a přidejte projekt do **StockQuoteAxis2EAR**. WASCE 2.1 je vybrána jako cílové běhové prostředí.

c) V sekci Konfigurace na stránce **Nový dynamický webový projekt** > **Upravit ...** > Zkontrolujte fasetu projektu webových služeb Axis2 . **Dynamický webový modul 2.5**, **Java 6.0**a **Implementace WASCE 1.2** jsou již zaškrtnuty. > **OK** > **Dokončit**. Odpovězte uživateli No do okna, ve kterém můžete otevřít perspektivu Java EE .

2. Importujte kód WSDL pro službu do pracovního prostoru a vygenerujte server proxy klienta:

V tomto příkladu dokument WSDL obsahuje vazbu služby HTTP a stane se cílem pro server proxy statického webového klienta. Před generováním serveru proxy klienta můžete upravit adresu URL ve vazbě webové služby tak, aby odkazovala na transport SOAP produktu WebSphere MQ . Server proxy statického webového klienta je poté služba, která je implementována pro transport protokolu SOAP produktu WebSphere MQ .

a) Spusťte průzkumník webových služeb: buď použijte ikonu na řádku s akcemi, nebo **Spustit** > **Spustit průzkumník webových služeb**.

b) Vyberte průzkumník WSDL klepnutím na ikonu WSDL v okně **Průzkumník webových služeb** > Klepněte na položku **WSDL-hlavní** v okně Navigator > Zadejte adresu URL souboru WSDL StockQuote> **Přejít**.
V tomto příkladu získáte WSDL přímo ze služby HTTP: `http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis?wsdl`

c) V Navigatorklepněte na řádek s adresou URL webové služby. V okně **Akce** klepněte na volbu **Importovat WSDL do pracovní plochy** > Vyberte **StockQuoteAxis2WebClient** jako **Projekt pracovní plochy** > Zadejte **Název souboru WSDL**, `StockQuoteAxisHTTP.wsdl` > **Přejít**.

d) Right-click **StockQuoteAxisHTTP.wsdl** > **Webové služby** > **Generovat klienta**. Zkontrolujte informace o konfiguraci na stránce průvodce následujícím způsobem: Server: IBM WASCE v2.1 Server, běhová komponenta webové služby: Apache Axis2, projekt klienta: StockQuoteAxis2WebClient, projekt klienta EAR: StockQuoteAxisEAR. Chcete-li opravit konfiguraci, klepněte na řádky, které jsou chybné.

e) Klepněte na tlačítko **Další** > ověřte nastavení generování kódu > **Dokončit**.

Všimněte si, že nový balík, `soap.server`, je vytvořen a obsahuje proxy, které požadujete.

3. Nakonfigurujte projekt pro spuštění přenosu WebSphere MQ pro SOAP jako přenos JMS.

Přenos WebSphere MQ pro SOAP poskytuje `transportSender`, ale ne `transportReceiver`. Jinými slovy, přenos WebSphere MQ pro protokol SOAP podporuje klienty Axis2 . Momentálně nepodporuje služby Axis2 .

a) V projektu **StockQuoteAxis2WebClient** klepněte pravým tlačítkem myši na položku `WebContent\WEB-INF\conf\axis2.xml` > **Otevřít pomocí ...** > **Editor XML**.

b) Vyhledejte poslední `transportSender` (směrem ke konci souboru) a vyhledejte komentář k rozhraní JMS `transportSender` > Right-click řádku > **Přidat před ...** > **transportSender**.

c) Klepněte pravým tlačítkem myši na **transportSender** > **Add Attribute** > **Name** > Right-click **transportSender** > **Add Attribute** > **Class**.

d) Klepněte pravým tlačítkem myši na **Název** > **Upravit atribut** > Zadejte **Hodnota:** `jms`

e) Right-click **Třída** > **Upravit atribut** > Zadejte **Hodnota:** `com.ibm.mq.axis2.transport.jms.WMQJMSTransportSender`. > Uložit.

f) Přidejte `com.ibm.mq.axis2.transport.jms.WMQJMSTransportSender` do cesty sestavení: Right-click **StockQuoteAxis2WebClient** > **Cesta sestavení** > **Konfigurovat cestu sestavení ...** > Klepněte na kartu **Knihovny** > **Přidat externí soubory JAR ...**. Vyberte všechny soubory JAR v produktu *MQ_INSTALLATION_PATH*`\java\lib` > **OK**.

MQ_INSTALLATION_PATH je adresář, kde je nainstalován produkt WebSphere MQ .

4. Vytvořte synchronního statického klienta, otestujte jej pomocí protokolu HTTP a potom převeďte server proxy ke spuštění statického klienta pomocí přenosu WebSphere MQ pro protokol SOAP.
   a) Right-click **Prostředky Java: src** > **Nový** > **Balík** > Název balíku `soap.client` > Dokončit
   b) Klepněte pravým tlačítkem myši na volbu **soap.client** > **New** > **Class** > Název třídy `SQA2StaticClient` > **Dokončit**.
   c) Nahraďte třídu následujícím kódem a poté klepněte na tlačítko **Uložit**.

   Obrázek 182. `SQA2DynamicClient.java`

```
package soap.client;
import soap.server.StockQuoteAxisServiceStub;
import soap.server.StockQuoteAxisServiceStub.GetQuote;
public class SQA2StaticClient {
  public static void main(String[] args) {
    try {
      StockQuoteAxisServiceStub stub = new StockQuoteAxisServiceStub();
      GetQuote request = new GetQuote();
      request.setSymbol("ibm");
      System.out.println("Response is: "
          + (stub.getQuote(request)).getGetQuoteReturn());
    } catch (Exception e) {
      System.out.println("Exception: " + e.getMessage());
      e.printStackTrace();
    }
  }
}
```

5. Otestujte klienta se službou Axis StockQuoteimplementovanou na WASCE a s přenosem WebSphere MQ pro SOAP.
   a) V Průzkumníku projektů klepněte pravým tlačítkem myši na položku **SQA2StaticClient** > **Spustit jako ...** > **Aplikace Java**.

      Výsledek, `Response is 55.25`, se zobrazí v pohledu Konzola. V pohledu Konzola můžete také vybrat okno konzoly WASCE a zobrazit výstup na serveru WASCE `StockQuoteAxis called with parameter: ibm`.
   b) Server proxy byl sestaven s adresou služby `http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis`, a tak statický klient volá službu spuštěnou na HTTP. Statický klient můžete změnit tak, aby volal službu pomocí přenosu WebSphere MQ pro protokol SOAP. Následující pokyny mění servisní adresu v produktu `StockQuoteAxisServiceStub.java` bez opětovného sestavení serveru proxy a konfigurují běhové parametry produktu `SQA2StaticClient` pro načtení `axis2.xml`. Nakonfigurujete `axis2.xml` pro konfiguraci Axis2 pro použití přenosu WebSphere MQ pro SOAP.
   c) Otevřete `StockQuoteAxisServiceStub.java` >
      Nahraďte dva výskyty prvku `http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis` řetězcem,

```
jms:/queue?destination=REQUESTAXIS@QM1
&connectionFactory=()
&initialContextFactory=com.ibm.mq.jms.Nojndi
&targetService=StockQuoteAxis
&replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE
```

   d) Spustíte-li produkt `SQA2StaticClient` nyní, vygeneruje výjimku, protože nenalezla `transportSender` konfigurované pro platformu JMS.
      Výjimka:

```
Exception: null java.lang.NullPointerException at
soap.server.StockQuoteAxisServiceStub.getQuote(StockQuoteAxisServiceStub.java:547)
at soap.client.SQA2StaticClient.main(SQA2StaticClient.java:11)
```

e) V Průzkumníku projektů klepněte pravým tlačítkem myši na **SQA2StaticClient** > **Spustit jako ...** > **Spustit konfigurace ...**. Přepněte na kartu **(x) = Argumenty** a do vstupní oblasti **Argumenty VM** zadejte cestu k souboru `axis2.conf` > **Použít** > **Spustit**.
Argument VM je: `-Daxis2.xml=${workspace_loc:StockQuoteAxis2WebClient/WebContent/WEB-INF/conf}/axis2.xml`. Případně můžete zadat standardní cestu ke konfiguračnímu souboru Axis2 .

f) Spusťte příkaz `SQA2StaticClient` znovu. V tomto spuštění používáte přenos WebSphere MQ pro SOAP. Potvrďte, že v konzole WASCE není žádný nový výstup. Otevřete konzolu nebo příkazové okno, které je přidruženo k modulu listener SimpleJava, a výstup je `StockQuoteAxis called with parameter: ibm`.

6. Vytvořte dynamického klienta pro přenos protokolu HTTP a WebSphere MQ pro protokol SOAP a otestujte jej.

a) Klepněte pravým tlačítkem myši na volbu **soap.client** > **New** > **Class** > Název třídy `SQA2DynamicClient` > **Dokončit**.

b) Nahraďte třídu následujícím kódem a poté klepněte na tlačítko **Uložit**.

```
package soap.client;
import soap.server.StockQuoteAxisServiceStub;
import soap.server.StockQuoteAxisServiceStub.GetQuote;
public class SQA2DynamicClient {
  public static void main(String[] args) {
    try {
      StockQuoteAxisServiceStub stub = new StockQuoteAxisServiceStub(
          "http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis");
      GetQuote request = new GetQuote();
      request.setSymbol("ibm");
      System.out.println("HTTP Sync: "
          + (stub.getQuote(request)).getGetQuoteReturn());
      stub = new StockQuoteAxisServiceStub(
          "jms:/queue?destination=REQUESTAXIS@QM1"
          + "&connectionFactory=()&initialContextFactory=com.ibm.mq.jms.Nojndi"
          + "&targetService=StockQuoteAxis&replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE");
       System.out.println("JMS sync: "
          + (stub.getQuote(request)).getGetQuoteReturn());
     } catch (Exception e) {
       System.out.println("Exception: " + e.getMessage());
       e.printStackTrace();
     }
   }
 }
```

c) Vytvořte konfiguraci spuštění pro produkt `SQA2DynamicClient.javaa` přidejte cestu k produktu `axis2.xml`:
`-Daxis2.xml=${workspace_loc:StockQuoteAxis2WebClient/WebContent/WEB-INF/conf}/axis2.xml`

d) Spusťte příkaz `SQA2DynamicClient`. Zkontrolujte výstup konzoly pro produkt `SQA2DynamicClient`, WASCE a **`SimpleJavaListener`**.

7. Vytvořte asynchronního klienta a přistupte k výsledku v obslužné rutině zpětného volání a v hlavním vláknu programu.

Proxy asynchronní klienta vytvořené pomocí průvodce webovou službou pro prostředí Eclipse Java EE IDE for Web Developers se liší od serverů proxy vytvořených produktem **`wsimport`**. Servery proxy produktu **`wsimport`** používají generické typy `Future`, `Responsea` `AsyncHandler` .

Průvodce webovými službami pro prostředí Eclipse Java EE IDE for Web Developers vytváří abstraktní třídu `StockQuoteAxisServiceCallbackHandler` . Je třeba rozšířit produkt `StockQuoteAxisServiceCallbackHandler` a vytvořit obslužnou rutinu zpětného volání.

a) Klepněte pravým tlačítkem myši na volbu **soap.client** > **New** > **Class** > Název třídy `SQA2CallbackHandler` > **Dokončit**.

b) Nahraďte třídu následujícím kódem.

```
package soap.client;
import soap.server.StockQuoteAxisServiceCallbackHandler;
import soap.server.StockQuoteAxisServiceStub.GetQuoteResponse;
```

```
public class SQA2CallbackHandler
    extends StockQuoteAxisServiceCallbackHandler {
  private boolean complete = false;
  SQA2CallbackHandler() {
    super();
    System.out.println("Callback constructor");
  }
  public void receiveResultgetQuote(GetQuoteResponse response) {
    System.out.println("Result in Callback " + response.getGetQuoteReturn());
    super.clientData = response;
    complete = true;
  }
  public boolean isComplete() {
    return complete;
  }
}
```

c) Klepněte pravým tlačítkem myši na volbu **soap.client** > **New** > **Class** > Název třídy `SQA2AsyncClient` > **Dokončit**.

d) Nahraďte třídu následujícím kódem.

Obrázek 183. `SQA2AsyncClient.java`

```
package soap.client;
import soap.server.StockQuoteAxisServiceStub;
import soap.server.StockQuoteAxisServiceStub.GetQuote;
import soap.server.StockQuoteAxisServiceStub.GetQuoteResponse;
import soap.server.StockQuoteAxisServiceCallbackHandler;
@SuppressWarnings("unused")
public class SQA2AsyncClient {
  public static void main(String[] args) {
    try {
      StockQuoteAxisServiceStub stub = new StockQuoteAxisServiceStub(
          "http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis");
      GetQuote request = new GetQuote();
      request.setSymbol("ibm");
      System.out.println("HTTP Sync: "
          + (stub.getQuote(request)).getGetQuoteReturn());
      SQA2CallbackHandler callback = new SQA2CallbackHandler();
      stub.startgetQuote(request, callback);
      do {
        System.out.println("Waiting for HTTP callback");
        Thread.sleep(2000);
      } while (!callback.isComplete());
      System.out.println("HTTP poll: "
          + ((GetQuoteResponse) (callback.getClientData()))
                .getGetQuoteReturn());
      stub = new StockQuoteAxisServiceStub(
          "jms:/queue?destination=REQUESTAXIS@QM1"
          + "&connectionFactory=()&initialContextFactory=com.ibm.mq.jms.Nojndi"
          + "&targetService=StockQuoteAxis&replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE");
      System.out.println("JMS Sync: "
          + (stub.getQuote(request)).getGetQuoteReturn());
      callback = new SQA2CallbackHandler();
      stub.startgetQuote(request, callback);
      while (!callback.isComplete()) {
        System.out.println("Waiting for JMS callback");
        Thread.sleep(2000);
      }
      System.out.println("JMS poll: "
        + ((GetQuoteResponse) (callback.getClientData())).getGetQuoteReturn());
    } catch (Exception e) {
        System.out.println("Exception: " + e.getMessage());
        e.printStackTrace();
     }
   }
 }
```

Výstup konzoly je následující:

```
HTTP Sync: 55.25
Callback constructor
Waiting for HTTP callback
Result in Callback 55.25
HTTP poll: 55.25
JMS Sync: 55.25
```

```
Callback constructor
Waiting for JMS callback
Result in Callback 55.25
JMS poll: 55.25
```

Ukázka klientů Axis2

Ukázkové servery proxy jsou generovány pomocí nástroje **wsimport** , který je zabalen s prostředím Java 6. K dispozici je šest vzorků:

1. DynamicProxyClientSync.java
2. DynamicProxyClientAsyncPolling.java
3. DynamicProxyClientAsyncCallback.java
4. DispatchClientSync.java
5. DispatchClientAsyncPolling.java
6. DispatchClientAsyncCallback.java

Ukázky klienta jsou generovány pro ukázku serveru Axis StockQuote. Vygenerujte WSDL příkazem **amqwdepoyWMQServer** zadáním přepínače -w , abyste vybrali styl `rpcLiteral` . Chcete-li vygenerovat proxy pro ukázky, použijte následující příkaz:

```
wsimport soap.server.StockQuoteAxis_Wmq.wsdl -d generated -keep -p com.ibm.mq.axis2.samples
```

Obrázek 184. DynamicProxyClientSync.java

```
package com.ibm.mq.axis2.samples;

import com.ibm.mq.axis2.samples.proxy.StockQuoteAxis;
import com.ibm.mq.axis2.samples.proxy.StockQuoteAxisService;

public class DynamicProxyClientSync {

  public static void main(String[] args) {
    try {
      System.out.println("Starting sample DynamicProxyClientSync");

      System.out.println("Creating proxy instance for service StockQuoteAxisService");
      StockQuoteAxisService stub = new StockQuoteAxisService();
      StockQuoteAxis service = stub.getSoapServerStockQuoteAxisWmq();

      System.out.println("Invoking getQuoteOneWay OneWay operation synchronously...");
      service.getQuoteOneWay("48");
      System.out.println(" > getQuoteOneWay has returned");

      System.out.println("Invoking getQuote Request Reply operation synchronously...");
      float result = service.getQuote("48");
      System.out.println(" > getQuote has returned result of " + result);

      System.out.println("End of sample");
    }
    catch (Exception fault) {
      // Identify the cause of the Axis Fault
      System.err.println(fault.toString());
      Throwable e = fault.getCause();
      for (int i = 1; e != null; i++) {
        // The toString method on an MQAxisException will cause the message, explanation and
user
        // action.
        System.err.println("Exception(" + i + "): " + e.toString());

        if (e.getCause() != null) {
          e = e.getCause();
        }
        else {
          break;
        }
      } // end of for loop
    } // end of catch block
```

```
  }
}
```

Obrázek 185. DynamicProxyClientAsyncPolling.java

```
package com.ibm.mq.axis2.samples;

import java.util.concurrent.CancellationException;

import javax.xml.ws.Response;

import com.ibm.mq.axis2.samples.proxy.StockQuoteAxis;
import com.ibm.mq.axis2.samples.proxy.StockQuoteAxisService;

public class DynamicProxyClientAsyncPolling {

  public static void main(String[] args) {
    try {
      System.out.println("Starting sample DynamicProxyClientAsyncPolling");

      System.out.println("Creating proxy instance for service StockQuoteAxisService");
      StockQuoteAxisService stub = new StockQuoteAxisService();
      StockQuoteAxis service = stub.getSoapServerStockQuoteAxisWmq();

      System.out
          .println("Invoking getQuoteAsync Request Reply operation asynchronously by
polling...");
      Response<Float> response = service.getQuoteAsync("49");

      /** Sleep main thread until response arrives **/
      System.out.println("Waiting for response to arrive...");
      while (!response.isDone()) {
        Thread.sleep(100);
      }
      System.out.println(" > Response received");

      /** Retrieve the result **/
      try {
        Float result = response.get();
        System.out.println(" > getQuoteAsync call has returned result of " + result);
      }
      catch (CancellationException ce) {
        // processing was cancelled via response.cancel()
      }

      System.out.println("End of sample");
    }
    catch (Exception fault) {
      // Identify the cause of the Axis Fault
      System.err.println(fault.toString());
      Throwable e = fault.getCause();
      for (int i = 1; e != null; i++) {
        // The toString method on an MQAxisException will cause the message, explanation and
user
        // action.
        System.err.println("Exception(" + i + "): " + e.toString());

        if (e.getCause() != null) {
          e = e.getCause();
        }
        else {
          break;
        }
      } // end of for loop
    } // end of catch block
  }
}
```

Obrázek 186. DynamicProxyClientAsyncCallback.java

```
package com.ibm.mq.axis2.samples;

import java.util.concurrent.Future;
```

```
import javax.xml.ws.AsyncHandler;
import javax.xml.ws.Response;

import com.ibm.mq.axis2.samples.proxy.StockQuoteAxis;
import com.ibm.mq.axis2.samples.proxy.StockQuoteAxisService;

public class DynamicProxyClientAsyncCallback implements AsyncHandler<Float> {

  public static void main(String[] args) {
    try {
      System.out.println("Starting sample DynamicProxyClientAsyncCallback");

      System.out.println("Creating proxy instance for service StockQuoteAxisService");
      StockQuoteAxisService stub = new StockQuoteAxisService();
      StockQuoteAxis service = stub.getSoapServerStockQuoteAxisWmq();

      DynamicProxyClientAsyncCallback handler = new DynamicProxyClientAsyncCallback();

      System.out
          .println("Invoking getQuoteAsync Request Reply operation asynchronously using a
callback...");
      Future<?> monitor = service.getQuoteAsync("50", handler);
      System.out.println(" > Invoke call has returned");

      /** Sleep main thread until handler has been notified **/
      System.out.println("Waiting for handler to be called...");
      while (!monitor.isDone()) {
        Thread.sleep(100);
      }

      System.out.println("End of sample");
    }
    catch (Exception fault) {
      // Identify the cause of the Axis Fault
      System.err.println(fault.toString());
      Throwable e = fault.getCause();
      for (int i = 1; e != null; i++) {
        // The toString method on an MQAxisException will cause the message, explanation and
user
        // action.
        System.err.println("Exception(" + i + "): " + e.toString());

        if (e.getCause() != null) {
          e = e.getCause();
        }
        else {
          break;
        }
      } // end of for loop
    } // end of catch block
  }

  public void handleResponse(Response<Float> response) {
    try {
      Float result = response.get();
      System.out.println(" > Async Handler has received a result of " + result);
    }
    catch (Exception fault) {
      // Identify the cause of the Axis Fault
      System.err.println("Exception in handleResponce");
      System.err.println(fault.toString());
      Throwable e = fault.getCause();
      for (int i = 1; e != null; i++) {
        // The toString method on an MQAxisException will cause the message, explanation and
user
        // action.
        System.err.println("Exception(" + i + "): " + e.toString());

        if (e.getCause() != null) {
          e = e.getCause();
        }
        else {
          break;
        }
      } // end of for loop
    } // end of catch block
  }

}
```

Obrázek 187. DispatchClientSync.java

```
package com.ibm.mq.axis2.samples;

import javax.xml.namespace.QName;
import javax.xml.soap.MessageFactory;
import javax.xml.soap.SOAPBody;
import javax.xml.soap.SOAPConstants;
import javax.xml.soap.SOAPElement;
import javax.xml.soap.SOAPEnvelope;
import javax.xml.soap.SOAPHeader;
import javax.xml.soap.SOAPMessage;
import javax.xml.soap.SOAPPart;
import javax.xml.ws.Dispatch;
import javax.xml.ws.Service;
import javax.xml.ws.soap.SOAPBinding;

public class DispatchClientSync {

  public static void main(String[] args) {
    try {
      System.out.println("Starting sample DispatchClientSync");

      String endpointUrl = "jms:/queue?destination=SOAPJ.demos@WMQSOAP.DEMO.QM&"
          + "connectionFactory=connectQueueManager(WMQSOAP.DEMO.QM)"
          +
"&initialContextFactory=com.ibm.mq.jms.Nojndi&targetService=soap.server.StockQuoteAxis.java";

      QName serviceName = new QName("soap.server.StockQuoteAxis_Wmq", "StockQuoteAxisService");
      QName portName = new QName("soap.server.StockQuoteAxis_Wmq",
"soap.server.StockQuoteAxis_Wmq");

      Service service = Service.create(serviceName);
      service.addPort(portName, SOAPBinding.SOAP11HTTP_BINDING, endpointUrl);

      /** Create a Dispatch instance from a service **/
      System.out.println("Creating dispatch instance for service StockQuoteAxisService");
      Dispatch<SOAPMessage> dispatch = service.createDispatch(portName, SOAPMessage.class,
          Service.Mode.MESSAGE);
      System.out.println(" > Dispatch instance created.");

      /************************************
       * Create OneWay SOAPMessage request.
       ************************************/
      MessageFactory mf = MessageFactory.newInstance(SOAPConstants.SOAP_1_1_PROTOCOL);

      System.out.println("\nCreating a OneWay SOAP Message");
      SOAPMessage request = mf.createMessage();

      /** Obtain the SOAPEnvelope and header and body elements **/
      SOAPPart part = request.getSOAPPart();
      SOAPEnvelope env = part.getEnvelope();
      SOAPHeader header = env.getHeader();
      SOAPBody body = env.getBody();

      /** Construct the message payload **/
      SOAPElement operation = body.addChildElement("getQuoteOneWay", "ns1",
          "soap.server.StockQuoteAxis_Wmq");
      SOAPElement value = operation.addChildElement("in0");
      value.addAttribute(new QName("https://www.w3.org/2001/XMLSchema-instance", "type"),
"string");
      value.addTextNode("XXX");
      request.saveChanges();
      System.out.println(" > SOAP Message created.");

      /** Invoke the service endpoint **/
      System.out.println("Invoking getQuoteOneWay OneWay operation synchronously...");
      dispatch.invokeOneWay(request);
      System.out.println(" > getQuoteOneWay call has returned");

      /****************************************
       * Create Request Reply SOAPMessage request.
       ****************************************/
      mf = MessageFactory.newInstance(SOAPConstants.SOAP_1_1_PROTOCOL);

      System.out.println("\nCreating a Request Reply SOAP Message");
      request = mf.createMessage();
```

```
      /** Obtain the SOAPEnvelope and header and body elements **/
      part = request.getSOAPPart();
      env = part.getEnvelope();
      header = env.getHeader();
      body = env.getBody();

      /** Construct the message payload **/
      operation = body.addChildElement("getQuote", "ns1", "soap.server.StockQuoteAxis_Wmq");
      value = operation.addChildElement("in0");
      value.addAttribute(new QName("https://www.w3.org/2001/XMLSchema-instance", "type"),
"string");
      value.addTextNode("XXX");
      request.saveChanges();
      System.out.println(" > SOAP Message created.");

      /** Invoke the service endpoint **/
      System.out.println("Invoking getQuote Request Reply operation synchronously...");
      SOAPMessage ans = dispatch.invoke(request);
      System.out.println(" > getQuote call has returned");

      /** Retrieve the result **/
      part = ans.getSOAPPart();
      env = part.getEnvelope();
      body = env.getBody();

      /** Define name of the SOAP folders we are interested in **/
      QName responseName = new QName("soap.server.StockQuoteAxis_Wmq", "getQuoteResponse");
      QName resultName = new QName("getQuoteReturn");

      /** Retrieve result from SOAP envelope **/
      System.out.println("Parsing SOAP response...");
      SOAPElement bodyElement = (SOAPElement) body.getChildElements(responseName).next();
      SOAPElement responseElement = (SOAPElement)
bodyElement.getChildElements(resultName).next();
      String message = responseElement.getValue();
      System.out.println(" > Response contains result of " + message);

      System.out.println("End of sample");
    }
    catch (Exception fault) {
      // Identify the cause of the Axis Fault
      System.err.println(fault.toString());
      Throwable e = fault.getCause();
      for (int i = 1; e != null; i++) {
        // The toString method on an MQAxisException will cause the message, explanation and
user
        // action.
        System.err.println("Exception(" + i + "): " + e.toString());

        if (e.getCause() != null) {
          e = e.getCause();
        }
        else {
          break;
        }
      } // end of for loop
    } // end of catch block
  }
}
```

Obrázek 188. DispatchClientAsyncPolling.java

```
package com.ibm.mq.axis2.samples;

import javax.xml.namespace.QName;
import javax.xml.soap.MessageFactory;
import javax.xml.soap.SOAPBody;
import javax.xml.soap.SOAPConstants;
import javax.xml.soap.SOAPElement;
import javax.xml.soap.SOAPEnvelope;
import javax.xml.soap.SOAPHeader;
import javax.xml.soap.SOAPMessage;
import javax.xml.soap.SOAPPart;
import javax.xml.ws.Dispatch;
import javax.xml.ws.Response;
import javax.xml.ws.Service;
```

```
import javax.xml.ws.soap.SOAPBinding;

public class DispatchClientAsyncPolling {

  public static void main(String[] args) {
    try {
      System.out.println("Starting sample DispatchClientAsyncPolling");

      String endpointUrl = "jms:/queue?destination=SOAPJ.demos@WMQSOAP.DEMO.QM&"
          + "connectionFactory=connectQueueManager(WMQSOAP.DEMO.QM)"
          +
"&initialContextFactory=com.ibm.mq.jms.Nojndi&targetService=soap.server.StockQuoteAxis.java";

      QName serviceName = new QName("soap.server.StockQuoteAxis_Wmq", "StockQuoteAxisService");
      QName portName = new QName("soap.server.StockQuoteAxis_Wmq",
"soap.server.StockQuoteAxis_Wmq");

      Service service = Service.create(serviceName);
      service.addPort(portName, SOAPBinding.SOAP11HTTP_BINDING, endpointUrl);

      /** Create a Dispatch instance from a service.* */
      System.out.println("Creating dispatch instance for service StockQuoteAxisService");
      Dispatch<SOAPMessage> dispatch = service.createDispatch(portName, SOAPMessage.class,
          Service.Mode.MESSAGE);
      System.out.println(" > Dispatch instance created.");

      /** Create SOAPMessage request. * */
      MessageFactory mf = MessageFactory.newInstance(SOAPConstants.SOAP_1_1_PROTOCOL);

      System.out.println("Creating a Request Reply SOAP Message");
      SOAPMessage request = mf.createMessage();

      /** Obtain the SOAPEnvelope and header and body elements **/
      SOAPPart part = request.getSOAPPart();
      SOAPEnvelope env = part.getEnvelope();
      SOAPHeader header = env.getHeader();
      SOAPBody body = env.getBody();

      /** Construct the message payload **/
      SOAPElement operation = body.addChildElement("getQuote", "ns1",
          "soap.server.StockQuoteAxis_Wmq");
      SOAPElement value = operation.addChildElement("in0");
      value.addAttribute(new QName("https://www.w3.org/2001/XMLSchema-instance", "type"),
"string");
      value.addTextNode("XXX");
      request.saveChanges();
      System.out.println(" > SOAP Message created.");

      /** Invoke the service endpoint **/
      System.out.println("Invoking getQuote Request Reply operation asynchronously by
polling...");
      Response<SOAPMessage> response = dispatch.invokeAsync(request);
      System.out.println(" > getQuote call has returned");

      /** Sleep main thread until response arrives **/
      System.out.println("Waiting for response to arrive...");
      while (!response.isDone()) {
        Thread.sleep(100);
      }
      System.out.println(" > Response received");

      /** retrieve the result **/
      SOAPMessage ans = response.get();
      part = ans.getSOAPPart();
      env = part.getEnvelope();
      body = env.getBody();

      /** Define name of the SOAP folders we are interested in **/
      QName responseName = new QName("soap.server.StockQuoteAxis_Wmq", "getQuoteResponse");
      QName resultName = new QName("getQuoteReturn");

      /** Retrieve result from SOAP envelope **/
      SOAPElement bodyElement = (SOAPElement) body.getChildElements(responseName).next();
      SOAPElement responseElement = (SOAPElement)
bodyElement.getChildElements(resultName).next();
      String message = responseElement.getValue();
      System.out.println(" > Response contains result of " + message);

      System.out.println("End of sample");

    }
    catch (Exception fault) {
```

```
      // Identify the cause of the Axis Fault
      System.err.println(fault.toString());
      Throwable e = fault.getCause();
      for (int i = 1; e != null; i++) {
        // The toString method on an MQAxisException will cause the message, explanation and
user
        // action.
        System.err.println("Exception(" + i + "): " + e.toString());

        if (e.getCause() != null) {
          e = e.getCause();
        }
        else {
          break;
        }
      } // end of for loop
    } // end of catch block
  }
}
```

Obrázek 189. DispatchClientAsyncCallback.java

```
package com.ibm.mq.axis2.samples;

import java.util.concurrent.Future;

import javax.xml.namespace.QName;
import javax.xml.soap.MessageFactory;
import javax.xml.soap.SOAPBody;
import javax.xml.soap.SOAPConstants;
import javax.xml.soap.SOAPElement;
import javax.xml.soap.SOAPEnvelope;
import javax.xml.soap.SOAPHeader;
import javax.xml.soap.SOAPMessage;
import javax.xml.soap.SOAPPart;
import javax.xml.ws.AsyncHandler;
import javax.xml.ws.Dispatch;
import javax.xml.ws.Response;
import javax.xml.ws.Service;
import javax.xml.ws.soap.SOAPBinding;

public class DispatchClientAsyncCallback implements AsyncHandler<SOAPMessage> {

  public static void main(String[] args) {
    try {
      System.out.println("Starting sample DispatchClientAsyncCallback");

      String endpointUrl = "jms:/queue?destination=SOAPJ.demos@WMQSOAP.DEMO.QM&"
          + "connectionFactory=connectQueueManager(WMQSOAP.DEMO.QM)"
          +
"&initialContextFactory=com.ibm.mq.jms.Nojndi&targetService=soap.server.StockQuoteAxis.java";

      QName serviceName = new QName("soap.server.StockQuoteAxis_Wmq", "StockQuoteAxisService");
      QName portName = new QName("soap.server.StockQuoteAxis_Wmq",
"soap.server.StockQuoteAxis_Wmq");

      Service service = Service.create(serviceName);
      service.addPort(portName, SOAPBinding.SOAP11HTTP_BINDING, endpointUrl);

      /** Create a Dispatch instance from a service.* */
      System.out.println("Creating dispatch instance for service StockQuoteAxisService");
      Dispatch<SOAPMessage> dispatch = service.createDispatch(portName, SOAPMessage.class,
          Service.Mode.MESSAGE);
      System.out.println(" > Dispatch instance created.");

      /** Create SOAPMessage request. * */
      MessageFactory mf = MessageFactory.newInstance(SOAPConstants.SOAP_1_1_PROTOCOL);

      System.out.println("Creating a Request Reply SOAP Message");
      SOAPMessage request = mf.createMessage();

      /** Obtain the SOAPEnvelope and header and body elements **/
      SOAPPart part = request.getSOAPPart();
      SOAPEnvelope env = part.getEnvelope();
      SOAPHeader header = env.getHeader();
      SOAPBody body = env.getBody();
```

```
      /** Construct the message payload. **/
      SOAPElement operation = body.addChildElement("getQuote", "ns1",
          "soap.server.StockQuoteAxis_Wmq");
      SOAPElement value = operation.addChildElement("in0");
      value.addAttribute(new QName("https://www.w3.org/2001/XMLSchema-instance", "type"),
"string");
      value.addTextNode("XXX");
      request.saveChanges();
      System.out.println(" > SOAP Message created.");

      /** Invoke the service endpoint. **/
      DispatchClientAsyncCallback handler = new DispatchClientAsyncCallback();

      System.out
          .println("Invoking getQuote Request Reply operation asynchronously using a
callback...");
      Future<?> monitor = dispatch.invokeAsync(request, handler);
      System.out.println(" > getQuote call has returned");

      /** Sleep main thread until handler has been notified **/
      System.out.println("Waiting for handler to be called...");
      while (!monitor.isDone()) {
        Thread.sleep(100);
      }

      System.out.println("End of sample");
    }
    catch (Exception fault) {
      // Identify the cause of the Axis Fault
      System.err.println(fault.toString());
      Throwable e = fault.getCause();
      for (int i = 1; e != null; i++) {
        // The toString method on an MQAxisException will cause the message, explanation and
user
        // action.
        System.err.println("Exception(" + i + "): " + e.toString());

        if (e.getCause() != null) {
          e = e.getCause();
        }
        else {
          break;
        }
      } // end of for loop
    } // end of catch block
  }

  public void handleResponse(Response<SOAPMessage> response) {
    try {
      // retrieve the result
      SOAPMessage ans = response.get();
      SOAPPart part = ans.getSOAPPart();
      SOAPEnvelope env = part.getEnvelope();
      SOAPBody body = env.getBody();

      /** Define name of the SOAP folders we are interested in **/
      QName responseName = new QName("soap.server.StockQuoteAxis_Wmq", "getQuoteResponse");
      QName resultName = new QName("getQuoteReturn");

      /** Retrieve result from SOAP envelope **/
      SOAPElement bodyElement = (SOAPElement) body.getChildElements(responseName).next();
      SOAPElement responseElement = (SOAPElement)
bodyElement.getChildElements(resultName).next();
      String result = responseElement.getValue();

      System.out.println(" > Async Handler has received a result of " + result);
    }
    catch (Exception fault) {
      // Identify the cause of the Axis Fault
      System.err.println("Exception in handleResponce");
      System.err.println(fault.toString());
      Throwable e = fault.getCause();
      for (int i = 1; e != null; i++) {
        // The toString method on an MQAxisException will cause the message, explanation and
user
        // action.
        System.err.println("Exception(" + i + "): " + e.toString());

        if (e.getCause() != null) {
          e = e.getCause();
        }
        else {
```

```
          break;
        }
      } // end of for loop
    } // end of catch block
  }

}
```

Související úlohy

Vývoj klienta JAX-RPC pro transport WebSphere pro protokol SOAP pomocí prostředí Eclipse
Vývoj klienta webové služby Axis 1.4 pro použití přenosu protokolu SOAP v produktu WebSphere MQ .

Vývoj klienta .NET 1 nebo 2 pro přenos WebSphere pro protokol SOAP pomocí produktu Microsoft Visual Studio 2008
Vytvořte klienta webové služby .NET 1 nebo 2, který má být spuštěn pomocí přenosu WebSphere MQ pro protokol SOAP.

Vývoj klienta .NET 1 nebo 2 pro přenos WebSphere pro protokol SOAP pomocí produktu Microsoft Visual Studio 2008

Vytvořte klienta webové služby .NET 1 nebo 2, který má být spuštěn pomocí přenosu WebSphere MQ pro protokol SOAP.

Než začnete

Vývoj klienta .NET 1 nebo 2 můžete spustit několika různými způsoby:

1. Pomocí produktu **amqwdeployWMQService** vygenerujte stuby klienta z webové služby a importujte je do produktu Visual Studio.
2. Pomocí produktu **java2wsdl** můžete generovat WSDL z implementace webové služby v jazyce Java a poté pomocí produktu `wsdl.exe`, který je dodáván s prostředím .NET, generovat stuby klienta.
3. Vygenerujte WSDL z implementace služby .NET `.asmx` služby pomocí produktu **amqswsdl**a poté použijte příkaz `wsdl.exe`.
4. Pokud jste vyvinuli a implementovali službu pro HTTP, použijte odkaz **Přidat webový odkaz ...** Průvodce v produktu Visual Studio ke konfiguraci klienta pro přístup ke službě HTTP. Upravte adresu URL na službu implementovanou v rámci přenosu WebSphere MQ pro protokol SOAP.

Úloha používá službu vyvinutou v produktu “Vývoj služby .NET 1 nebo 2 pro přenos WebSphere MQ pro SOAP pomocí produktu Microsoft Visual Studio 2008” na stránce 932.

Informace o této úloze

Postupujte takto, chcete-li vytvořit přenos klienta .NET 1 nebo 2 pro protokol HTTP a produkt WebSphere MQ pro protokol SOAP.

Postup

1. Vytvořte aplikaci konzoly klienta a upravte ji tak, aby vyvolala webovou službu HTTP StockQuote .
   a) Klepněte pravým tlačítkem myši na volbu **Řešení 'StockQuoteDotNet'** v **Průzkumníku řešení** > Přidat ... > Nový projekt. Vyberte typ projektu **C#** , **.Rámec NET 2.0**a **Aplikace konzoly**. Pojmenujte projekt `StockQuoteClientDotNet` > **OK**
   b) Klepněte pravým tlačítkem myši na volbu **Řešení 'StockQuoteDotNet'** v **Průzkumníku řešení** > Přidat ... > Nový projekt. Vyberte typ projektu **C#** , **.Rámec NET 2.0**a **Aplikace konzoly**. Pojmenujte projekt `StockQuoteClientDotNet` > **OK**
   c) Right-click **StockQuoteClientDotNet** > **Nastavit jako projekt spuštění**.
   d) Right-click **StockQuoteClientDotNet** > **Přidat webový odkaz ...** > Procházet a hledat webové služby v tomto řešení > Vyberte **StockQuote** > **Přidat odkaz**. Všimněte si, že jste přidali webový odkaz na lokálního hostitele a nový konfigurační soubor `app.config`.

e) V průzkumníku řešení změňte název aplikace konzoly z `Program.cs` na `StockQuoteClientDotNet.cs` > Klepnutím na tlačítko **OK** změníte všechna použití `Program.cs` na `StockQuoteClientDotNet.cs`.

f) Nahraďte obsah souboru `StockQuoteClientDotNet.cs` kódem v souboru Obrázek 190 na stránce 964.

```
using System;
using StockQuoteClientDotNet.localhost;
namespace StockQuoteClientDotNet {
  class StockQuoteClientDotNet {
    static void Main(string[] args) {
      try {
        StockQuote stockobj = new StockQuote();
        Console.WriteLine("http reply is: "
          + stockobj.getNonInlineQuote("http request");
      }
      catch (System.Exception e) {
        Console.WriteLine("Exception thrown: " + e.ToString());
      }
      Console.ReadLine();
    }
  }
}
```

Obrázek 190. Obslužný program HTTP StockQuoteClientDotNet

g) Spusťte produkt StockQuoteClientDotNet na test vůči službě `StockQuote.asmx` :

i) Stiskněte tlačítko **F5**, klepněte na zelenou šipku v řádku s akcemi nebo na tlačítko **Ladit** > **Spustit ladění (F5)**.

Pokud se projekt StockQuoteDOtNet nachází ve stejném řešení, spustí se automaticky. V opačném případě je třeba nejprve spustit službu.

Příkazové okno s výsledky se otevře za pracovním prostorem. Příkaz `Console.ReadLine();` zabrání, aby se zavřel, dokud nestisknete klávesu **Enter**.

Tip: Ujistěte se, že `StockQuote.asmx` je počáteční stránka v projektu `StockQuoteDotNet` .

2. Upravte StockQuoteClientDotNet tak, aby volala službu `StockQuote.asmx` pomocí přenosu WebSphere MQ pro SOAP.

a) Přidejte řádky zobrazené tučným písmem na klienta.

```
using System;
using StockQuoteClientDotNet.localhost;
namespace StockQuoteClientDotNet {
  class StockQuoteClientDotNet {
    static void Main(string[] args) {
      try {
        IBM.WMQSOAP.Register.Extension();
        StockQuote stockobj = new StockQuote();
        Console.WriteLine("http reply is: "
          + stockobj.getNonInlineQuote("http request");
        stockobj.Url = "jms:/queue?"
          + "initialContextFactory=com.ibm.mq.jms.Nojndi"
          + "&connectionFactory=()&destination=REQUESTDOTNET@QM1"
          + "&targetService=StockQuote.asmx";
        Console.WriteLine("jms reply is: "
          + stockobj.getNonInlineQuote("jms request"));
      }
      catch (System.Exception e) {
        Console.WriteLine("Exception thrown: " + e.ToString());
      }
      Console.ReadLine();
    }
  }
}
```

Obrázek 191. Upravený program StockQuoteClientDotNet

Případně upravte výchozí adresu URL. Otevřete nabídku **StockQuoteClientDotNet** > **Properties** > **Settings.settings** a změňte hodnotu vlastnosti `StockQuoteClientDotNet_localhost_StockQuote` na přenos WebSphere MQ pro adresu URL protokolu SOAP.

b) Přidat odkaz na `amqsoap.dll`

i) V projektu **StockQuoteClientDotNet** v **Průzkumníku řešení**klepněte pravým tlačítkem myši na položku **Odkazy** > **Přidat odkaz ...** > Klepněte na kartu **Procházet** > přejděte k produktu *MQ_INSTALLATION_PATH*\bin > Vyberte volbu **amqsoap.dll** > **OK**. *MQ_INSTALLATION_PATH* je adresář, kde je nainstalován produkt WebSphere MQ .

3. Otestujte klienta se službou `StockQuote.asmx` pomocí přenosu WebSphere MQ pro SOAP.

a) Otevřete příkazové okno v adresáři projektu StockQuoteDotNet : `.\StockQuoteDotNet\StockQuoteDotNet` > Ověřte, že `.bin\StockQuoteDotNet.dll` existuje. Pokud tomu tak není, znovu sestavte řešení.

b) Napište příkaz **amqwRegisterdotNet**.

amqwRegisterdotNet je třeba spustit pouze jednou při instalaci.

c) Pokud jste spustili příkaz **amqwdeployWMQServer** s parametrem `genAsmxWMQBits`, spusťte modul listener SOAP rozhraní .NET: `generated\server\startWMQNListener`

d) Případně spusťte modul listener přímo:

```
amqwSOAPNETListener -u "jms:/queue?
destination=REQUESTDOTNET@QM1
&connectionFactory=()&initialContextFactory=com.ibm.mq.jms.Nojndi
&targetService=StockQuote.asmx&replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE"
-w C:\IBM\ID\StockQuoteDotNet\StockQuoteDotNet -n 10
```

4. V produktu Visual Studio 2008 stiskněte klávesu **F5** pro spuštění příkazu StockQuoteClientDotNet.

Klienti platformy .NET Framework 1 a .NET Framework 2 Web

Ukázkové klienty prostředí .NET poskytnuté s přenosem WebSphere MQ pro SOAP používají vygenerované stuby pro volání ukázkových služeb Axis a .NET.

V případě klientů .NET Framework 1 a .NET Framework 2 poskytuje produkt WebSphere MQ přístup k webovým službám pomocí klientů .NET. Příkaz **amqwdeployWMQService** má volbu `genProxiestoDotNet`, která generuje stuby klienta .NET Framework 1 nebo .NET Framework 2 pro webovou službu. Můžete také použít stuby klienta generované nástrojem .NET **wsdl** nebo produktem Microsoft Visual Studio 2005 nebo 2008.

Ukázky klientů webové služby .NET Framework 1 a .NET jsou nainstalovány v produktu *MQ_INSTALLATION_PATH*\tools\soap\samples\dotnet. *MQ_INSTALLATION_PATH* je adresář, kde je nainstalován produkt WebSphere MQ .

SQVB2Axis.vb
SQVB2Axis.vb, Obrázek 192 na stránce 966, je klient Visual Basic k volání služby **StockQuoteAxisService** .

SQVB2DotNet.vb
QVB2DotNet.vb, Obrázek 193 na stránce 966, je klient Visual Basic k volání služby **StockQuoteDotNet** .

SQCS2Axis.cs
SQCS2Axis.cs, Obrázek 194 na stránce 966, je klient C# pro volání služby **StockQuoteAxisService** . Adresu URL služby můžete přepsat zadáním adresy URL na příkazovém řádku.

SQCS2DotNet.cs
SQCS2DotNet.cs, Obrázek 195 na stránce 967, je klient C# pro volání služby **StockQuoteDotNet** . Adresu URL služby můžete přepsat zadáním adresy URL na příkazovém řádku.

```
Module SQVB2Axis
   Function Main(ByVal CmdArgs() As String) As Integer
      IBM.WMQSOAP.Register.Extension()
      Dim obj As New StockQuoteAxisService()
      Dim res As Single = obj.getQuote("fromcs")
      System.Console.WriteLine("SQVB2Axis: reply is: '{0}'", res)
   End Function
End Module
```

Obrázek 192. SQVB2Axis

```
Module SQVB2DotNet
   Function Main(ByVal CmdArgs() As String) As Integer
      IBM.WMQSOAP.Register.Extension()
      Dim obj as new StockQuoteDotNet()
      Dim res as Single = obj.getQuote("fromcs")
      System.Console.WriteLine("SQVB2DotNet: reply is: '{0}'", res)
   End Function
End Module
```

Obrázek 193. SQVB2DotNet

```
using System;
class SQCS2Axis {
   [STAThread]
   static void Main(string[] args) {
      try {
         IBM.WMQSOAP.Register.Extension();
         StockQuoteAxisService stockobj = new StockQuoteAxisService();
         if (args.GetLength(0) >= 1)
            stockobj.Url = args[0];
         System.Single res = stockobj.getQuote("XXX");
         Console.WriteLine("SQCS2Axis RPC reply is: " + res);
      }
      catch (System.Exception e) {
         Console.WriteLine("\n>>> EXCEPTION WHILE RUNNING SQCS2Axis DEMO <<<\n"
            + e.ToString());
      }
   }
}
```

Obrázek 194. SQCS2Axis

```
using System;
class SQCS2DotNet {
   [STAThread]
   static void Main(string[] args) {
      try {
         IBM.WMQSOAP.Register.Extension();
         StockQuoteDotNet stockobj = new StockQuoteDotNet();
         if (args.GetLength(0) >= 1)
            stockobj.Url = args[0];
         System.Single res = stockobj.getQuote("XXX");
         Console.WriteLine("RPC reply is: " + res);
         if (args.GetLength(0) == 0) {
            res = stockobj.getQuoteDOC("XXX");
            Console.WriteLine("DOC reply is: " + res);
         }
      }
      catch (System.Exception e) {
         Console.WriteLine("\n>>> EXCEPTION WHILE RUNNING SQCS2DotNet DEMO <<<\n"
            + e.ToString());
      }
   }
}
```

Obrázek 195. SQCS2DotNet

Související úlohy

Vývoj klienta JAX-RPC pro transport WebSphere pro protokol SOAP pomocí prostředí Eclipse
Vývoj klienta webové služby Axis 1.4 pro použití přenosu protokolu SOAP v produktu WebSphere MQ .

Vyvíjení klienta JAX-WS pro přenos WebSphere pro protokol SOAP pomocí prostředí Eclipse
Vytvořte klienta webové služby Axis2 pro použití přenosu protokolu SOAP produktu WebSphere MQ . Ukázkový klient Axis2 poskytovaný s přenosem WebSphere MQ pro protokol SOAP je uveden v seznamu a příkaz **wsimport** použitý ke generování serverů proxy.

Implementace webových služeb pomocí přenosu WebSphere MQ pro SOAP

Nasaďte webovou službu na jeden z mnoha různých serverových prostředí a připojte se k němu pomocí přenosu WebSphere MQ pro SOAP.

Než začnete

Vytvořte webovou službu a otestujte ji pomocí protokolu SOAP prostřednictvím protokolu HTTP v cílovém prostředí.

Informace o této úloze

Můžete implementovat webovou službu pro provoz s přenosem WebSphere MQ pro protokol SOAP v řadě různých běhových prostředí SOAP. Službu můžete implementovat na Axis 1.4 pouze s použitím softwaru nainstalovaného s produktem WebSphere MQ. V případě jiných běhových prostředí musíte instalovat další software.

Nejste omezeni na spuštění přenosu produktu WebSphere MQ pro protokol SOAP na servery, pro které jsou pokyny k implementaci. Postupujte podle pokynů pro implementaci služby do jednoho z uvedených prostředí.

Poznámka: Některá integrovaná prostředí nabízejí protokol SOAP prostřednictvím rozhraní JMS s použitím doporučených vazeb JMS SOAP organizace W3C a také přenosu WebSphere MQ pro vazbu SOAP. Verze produktu WebSphere MQ, včetně 7.0.1.2, podporují pouze přenos dat WebSphere MQ pro vazbu SOAP. Počínaje verzí 7.0.1.3 můžete implementovat klienty Axis2 pomocí identifikátoru URI, který odpovídá doporučenému doporučení W3C pro protokol SOAP prostřednictvím rozhraní JMS. Viz výukový program Vývoj aplikací webových služeb SOAP/JMS JAX-WS s produkty WebSphere Application Server V7 a Rational Application Developer V7.5.

Implementace služby na Axis 1.4 k použití pro přenos WebSphere pro protokol SOAP pomocí produktu amqwdeployWMQService

Implementujte službu Axis 1.4 do přenosu WebSphere MQ pro protokol SOAP vytvořením adresáře implementace, spuštěním příkazu **amqwdeployWMQService** a spuštěním modulu listener Axis 1.4 .

Než začnete

1. Postupujte podle pokynů pro instalaci přenosu produktu WebSphere MQ pro protokol SOAP
2. Ověřte instalaci a prostředí pomocí příkazu **runivt** .
3. Chcete-li znovu nasadit službu:
   a. Odstraňte podadresář `./generated` a všechny jeho podadresáře.
   b. Odeberte požadavky z cílové fronty a odstraňte ji.
   c. Pokračujte podle pokynů z kroku “2” na stránce 968.

Informace o této úloze

Tyto pokyny slouží k první implementaci služby Axis 1.4 . Chcete-li restartovat službu Axis 1.4 , spusťte znovu modul listener protokolu SOAP 1.4 : krok “11” na stránce 969.

Chcete-li implementovat novou službu Axis 1.4 do přenosu WebSphere MQ pro SOAP, postupujte podle následujících pokynů:

Postup

1. Vytvořte adresář *deployDir* , který bude obsahovat soubory implementace.
   Obslužný program implementace vyžaduje, aby byla každá služba implementována ze samostatného adresáře.
2. Otevřete příkazové okno na systému Windowsnebo příkazový shell pomocí systému X Window System na systému UNIX and Linux , v *deployDir* ke spuštění **amqwdeployWMQService**.
3. Spuštěním příkazu **amqwsetcp** nastavte cestu ke třídě.
   Prostředí JRE a sada JDK musí být v cestě ke třídě, ve verzi 5.0 nebo novější a na stejné úrovni verze.
4. Zkopírujte zdroj třídy *className*`.java`do *deployDir* .
5. Okopírujte všechny zdrojové soubory Java ve stejném balíku jako *className* do *deployDir*/*packageName*, kde *packageName* je adresářový strom odpovídající názvu balíku.
6. Spustit **javac** *packageName*`.`*className*`.`
   Možná budete muset přidat cestu k aktuálnímu adresáři "`.`", nebo do adresáře *packageName* pro **javac** , abyste našli ostatní třídy.
7. Vytvořit WSDL osy pro službu:

```
amqwdeployWMQService -f packageName.className.java -c genAxisWsdl
-v -u  "jms:/queue?destination=queueName
&initialContextFactory=com.ibm.mq.jms.Nojndi
&connectionFactory=(connectQueueManager(QmgrName)binding(auto))"
```

8. Vytvořte prostředky produktu WebSphere MQ pro danou službu:

```
amqwdeployWMQService -f packageName.className.java -c genAxisWMQBits
-v -u  "jms:/queue?destination=queueName
&initialContextFactory=com.ibm.mq.jms.Nojndi
&connectionFactory=(connectQueueManager(QmgrName)binding(auto))"
```

Tip:

Chcete-li nastavit nového správce front a prostředky, které potřebuje, abyste mohli provádět vývoj a testování, spusťte produkt **setupWMQSOAP**.

Chcete-li nastavit nového správce front jako výchozí, vezměte kopii produktu **setupWMQSOAP** z adresáře *WMQ install directory*\tools\soap\samples a do řádku přidejte parametr -q .

```
call :try -q crtmqm %QMGR%
```

9. Vytvořte modul listener Axis a implementujte službu:

```
amqwdeployWMQService -f packageName.className.java -c AxisDeploy
-v -u  "jms:/queue?destination=queueName
&initialContextFactory=com.ibm.mq.jms.Nojndi
&connectionFactory=(connectQueueManager(QmgrName)binding(auto))"
```

10. Pokud potřebujete generovat kód WSDL pro službu, generovat stuby klienta nebo servery proxy klienta, spusťte **amqwdeployWMQService** s jedním z následujících parametrů:

    - genAsmxWsdl
    - genAxisWsdl
    - genProxiesToDotNet
    - genProxiestoOsa

    Poznámka: Před generováním serverů proxy je třeba vygenerovat kód WSDL. Volba AllAxis selže, pokud není proměnná CLASSPATH nastavena pro nalezení všech tříd, které jsou importovány ke kompilaci *className*.java. Existuje-li více souborů Java v balíku obsahujícím *className*.java, musíte je nejprve zkompilovat pomocí produktu **javac**. **amqwdeployWMQService** -f *packageName*.*className*.java -c CompileJava kompiluje pouze *className*.java.

11. Spusťte generovaný modul listener Axis.

```
.\generated\server\startWMQJListener.cmd
```

Související úlohy

Implementace služby pro službu .NET Framework 1 nebo 2 pro použití přenosu WebSphere MQ pro SOAP
Implementujte službu .NET Framework 1 nebo 2 do transportu produktu WebSphere MQ pro SOAP. Vytvořte adresář implementace, spusťte příkaz **amqwdeployWMQService** a spusťte modul listener .NET.

Implementace služby na server CICS Transaction Server pro použití přenosu WebSphere pro protokol SOAP
Přenos produktu WebSphere MQ pro SOAP je integrován do podpory webových služeb CICS Transaction Server 4.1 .

Implementace služby na server WebSphere Application Server pro použití přenosu WebSphere pro protokol SOAP
Přenos protokolu WebSphere MQ pro SOAP je integrován do sběrnice pro integraci služeb na serveru WebSphere Application Server.

Konfigurace serveru WebSphere Application Server pro použití W3C SOAP přes JMS
Webová služba, která je svázána s doporučujícím doporučením W3C pro protokol SOAP prostřednictvím rozhraní JMS, musí být spuštěna v kontejneru EJB aplikačního serveru Java EE . Tato úloha je krokem 1 připojení klienta webové služby Axis2 a webové služby implementované na serveru WebSphere Application Server pomocí protokolu W3C protokolu SOAP prostřednictvím rozhraní JMS. Nakonfigurujte prostředky WebSphere MQ a WebSphere Application Server pro vývoj a implementaci vazby webových služeb na rozhraní W3C SOAP prostřednictvím rozhraní JMS jako přenosu.

Implementace služby na koncový bod služby WebSphere ESB a Process Server pro použití přenosu WebSphere pro protokol SOAP
Přenos produktu WebSphere MQ pro SOAP není přímo podporován produkty WebSphere ESB a Process Server. Musíte nakonfigurovat vlastní export.

Implementace služby pro službu .NET Framework 1 nebo 2 pro použití přenosu WebSphere MQ pro SOAP

Implementujte službu .NET Framework 1 nebo 2 do transportu produktu WebSphere MQ pro SOAP. Vytvořte adresář implementace, spusťte příkaz **amqwdeployWMQService** a spusťte modul listener .NET.

Než začnete

1. Postupujte podle pokynů pro instalaci přenosu produktu WebSphere MQ pro protokol SOAP
2. Ověřte instalaci a prostředí pomocí příkazu **runivt** .
3. Musí být nastavena cesta k souborům rámce .NET `wsdl.exe` a `csc.exe` . Kopie `wsdl.exe` a `csc.exe` identifikované proměnnou PATH musí být na stejné úrovni rámce .NET. Pokud máte nainstalováno více rámců .NET nebo používáte produkt Visual Studio, zkontrolujte pečlivě proměnnou PATH .
4. Chcete-li znovu nasadit službu:
   a. Odstraňte podadresář `./generated` a všechny jeho podadresáře.
   b. Odeberte požadavky z cílové fronty a odstraňte ji.
   c. Pokračujte podle pokynů z kroku “2” na stránce 970.

Informace o této úloze

Tyto pokyny mají implementovat službu .NET pro první použití. Chcete-li restartovat službu .NET, spusťte znovu modul listener rozhraní .NET SOAP, krok “9” na stránce 971.

Chcete-li implementovat novou službu .NET Framework 1 nebo .NET Framework 2 do transportu produktu WebSphere MQ pro SOAP, postupujte podle následujících pokynů:

Postup

1. Vytvořte adresář *deployDir* , který bude obsahovat soubory implementace.
   Obslužný program implementace vyžaduje, aby byla každá služba implementována ze samostatného adresáře.
2. Otevřete příkazové okno v produktu *deployDir* a spusťte příkaz **amqwdeployWMQService**.

```
C:\IBM\ID\QuoteClient>
```

3. Spuštěním příkazu **amqwsetcp** nastavte cestu ke třídě.
   Cesta ke třídě je potřebná pouze pro klienty Axis.
4. Zkopírujte službu .NET, *className*`.asmx`, do *deployDir*
5. Sestavte implementaci služby do knihovny (`.dll`).
   Vložená implementace služby je v produktu *className*`.asmx`. Implementace služby za provozu by mohla být *className*`.asmx.cs`.
   Obrázek 196 na stránce 970 uvádí příklad příkazu k sestavení služby .NET Framework V2 jako knihovny.

```
c:\WINDOWS\Microsoft.NET\Framework\v3.5\Csc.exe /noconfig /nowarn:1701,1702
/errorreport:prompt /warn:4 /define:TRACE
/reference:c:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\System.configuration.dll
/reference:c:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\System.Data.dll
/reference:c:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\System.dll
/reference:c:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\System.Drawing.dll
/reference:c:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\System.Web.dll
/reference:c:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\System.Web.Services.dll
/reference:c:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\System.Xml.dll
/debug:pdbonly /filealign:512 /optimize+
/out:obj\Quote.dll /target:library Properties\AssemblyInfo.cs Quote.asmx.cs
```

Obrázek 196. Příkaz sestavení pro službu .NET Framework V2

6. Zkopírujte *className*`.dll` do *deployDir*`\bin`.
7. Nastavte prostředky produktu WebSphere MQ a vytvořte modul listener vyžadovaný pro službu:

```
amqwdeployWMQService -f className.asmx -c genAsmxWMQBits
-v -u  "jms:/queue?destination=queueName
&initialContextFactory=com.ibm.mq.jms.Nojndi
&connectionFactory=(connectQueueManager(QmgrName)binding(auto))
&targetService=className.asmx"
```

8. Pokud potřebujete generovat kód WSDL pro službu, generovat stuby klienta nebo servery proxy klienta, spusťte **amqwdeployWMQService** s jedním z následujících parametrů:
   - genAsmxWsdl
   - genAxisWsdl
   - genProxiesToDotNet
   - genProxiestoOsa

   Poznámka: Před generováním serverů proxy je třeba vygenerovat kód WSDL.
9. Spusťte vygenerovaný modul listener .NET.

```
.\generated\server\startWMQNListener.cmd
```

Související úlohy

Implementace služby na Axis 1.4 k použití pro přenos WebSphere pro protokol SOAP pomocí produktu amqwdeployWMQService
Implementujte službu Axis 1.4 do přenosu WebSphere MQ pro protokol SOAP vytvořením adresáře implementace, spuštěním příkazu **amqwdeployWMQService** a spuštěním modulu listener Axis 1.4 .

Implementace služby na server CICS Transaction Server pro použití přenosu WebSphere pro protokol SOAP
Přenos produktu WebSphere MQ pro SOAP je integrován do podpory webových služeb CICS Transaction Server 4.1 .

Implementace služby na server WebSphere Application Server pro použití přenosu WebSphere pro protokol SOAP
Přenos protokolu WebSphere MQ pro SOAP je integrován do sběrnice pro integraci služeb na serveru WebSphere Application Server.

Konfigurace serveru WebSphere Application Server pro použití W3C SOAP přes JMS
Webová služba, která je svázána s doporučujícím doporučením W3C pro protokol SOAP prostřednictvím rozhraní JMS, musí být spuštěna v kontejneru EJB aplikačního serveru Java EE . Tato úloha je krokem 1 připojení klienta webové služby Axis2 a webové služby implementované na serveru WebSphere Application Server pomocí protokolu W3C protokolu SOAP prostřednictvím rozhraní JMS. Nakonfigurujte prostředky WebSphere MQ a WebSphere Application Server pro vývoj a implementaci vazby webových služeb na rozhraní W3C SOAP prostřednictvím rozhraní JMS jako přenosu.

Implementace služby na koncový bod služby WebSphere ESB a Process Server pro použití přenosu WebSphere pro protokol SOAP
Přenos produktu WebSphere MQ pro SOAP není přímo podporován produkty WebSphere ESB a Process Server. Musíte nakonfigurovat vlastní export.

Implementace služby na server CICS Transaction Server pro použití přenosu WebSphere pro protokol SOAP

Přenos produktu WebSphere MQ pro SOAP je integrován do podpory webových služeb CICS Transaction Server 4.1 .

Než začnete

Použijte stejné nástroje, které se mají vyvinout pro klienta nebo službu pro produkt WebSphere MQ, jak byste se měli vyvíjet pro HTTP. CICS má nástroje odpovídající **Java2wsdl** a **wsdl2Java**:

- **DFHWS2LS** přebírá popis webové služby jako výchozí bod. Používá popisy zpráv a datové typy použité v těchto zprávách k vytvoření datových struktur jazykových dat na vysoké úrovni. Můžete je použít ve strukturách v aplikačních programech napsaných v různých jazycích.

- Produkt **DFHLS2WS** jako výchozí bod používá datovou strukturu jazyka vyšší úrovně. Používá strukturu pro konstrukci popisu webových služeb, který obsahuje popisy zpráv. Vytvoří také schémata pro zprávy ze struktury dat jazyka.

Postupujte podle pokynů Vytvoření webové služby v dokumentaci k produktu CICS a vytvořte webovou službu.

Informace o této úloze

Postupujte podle pokynů v části Konfigurace systému CICS pro použití přenosu WebSphere MQ v dokumentaci produktu CICS . Pomocí pokynů můžete implementovat webovou službu do transportu produktu WebSphere MQ pro SOAP.

Související úlohy

Implementace služby na Axis 1.4 k použití pro přenos WebSphere pro protokol SOAP pomocí produktu amqwdeployWMQService
Implementujte službu Axis 1.4 do přenosu WebSphere MQ pro protokol SOAP vytvořením adresáře implementace, spuštěním příkazu **amqwdeployWMQService** a spuštěním modulu listener Axis 1.4 .

Implementace služby pro službu .NET Framework 1 nebo 2 pro použití přenosu WebSphere MQ pro SOAP
Implementujte službu .NET Framework 1 nebo 2 do transportu produktu WebSphere MQ pro SOAP. Vytvořte adresář implementace, spusťte příkaz **amqwdeployWMQService** a spusťte modul listener .NET.

Implementace služby na server WebSphere Application Server pro použití přenosu WebSphere pro protokol SOAP
Přenos protokolu WebSphere MQ pro SOAP je integrován do sběrnice pro integraci služeb na serveru WebSphere Application Server.

Konfigurace serveru WebSphere Application Server pro použití W3C SOAP přes JMS
Webová služba, která je svázána s doporučujícím doporučením W3C pro protokol SOAP prostřednictvím rozhraní JMS, musí být spuštěna v kontejneru EJB aplikačního serveru Java EE . Tato úloha je krokem 1 připojení klienta webové služby Axis2 a webové služby implementované na serveru WebSphere Application Server pomocí protokolu W3C protokolu SOAP prostřednictvím rozhraní JMS. Nakonfigurujte prostředky WebSphere MQ a WebSphere Application Server pro vývoj a implementaci vazby webových služeb na rozhraní W3C SOAP prostřednictvím rozhraní JMS jako přenosu.

Implementace služby na koncový bod služby WebSphere ESB a Process Server pro použití přenosu WebSphere pro protokol SOAP
Přenos produktu WebSphere MQ pro SOAP není přímo podporován produkty WebSphere ESB a Process Server. Musíte nakonfigurovat vlastní export.

Implementace služby na server WebSphere Application Server pro použití přenosu WebSphere pro protokol SOAP

Přenos protokolu WebSphere MQ pro SOAP je integrován do sběrnice pro integraci služeb na serveru WebSphere Application Server.

Než začnete

Použijte produkt Rational Application Developer, WebSphere Integration Developer nebo sadu nástrojů webových služeb pro vývoj webové služby.

Informace o této úloze

Pomocí následujících pokynů implementujte službu pomocí přenosu WebSphere MQ pro protokol SOAP jako přenos SOAP na serveru WebSphere Application Server.

Postup

1. Nakonfigurujte produkt WebSphere MQ jako poskytovatele systému zpráv JMS pro sběrnici pro integraci služeb na serveru WebSphere Application Server.
2. Nakonfigurujte prostředky produktu WebSphere MQ vyžadované službou.

3. Postupujte podle pokynů v dokumentaci produktu WebSphere Application Server Network Deployment v části Konfigurace prostředků JMS pro synchronní protokol SOAP přes koncový modul listener JMS.

   Pro ostatní platformy WebSphere Application Server existují odpovídající pokyny.
4. Upravte identifikátor URI služby tak, aby odpovídal transportu produktu WebSphere MQ pro identifikátor URI protokolu SOAP.
5. Implementujte službu na server WebSphere Application Server.

Jak pokračovat dále

Implementujte službu pomocí protokolu HTTP jako transport, aby se klienti mohli dotazovat na službu a přijímat WSDL v odezvě.

Související úlohy

Implementace služby na Axis 1.4 k použití pro přenos WebSphere pro protokol SOAP pomocí produktu amqwdeployWMQService
Implementujte službu Axis 1.4 do přenosu WebSphere MQ pro protokol SOAP vytvořením adresáře implementace, spuštěním příkazu **amqwdeployWMQService** a spuštěním modulu listener Axis 1.4 .

Implementace služby pro službu .NET Framework 1 nebo 2 pro použití přenosu WebSphere MQ pro SOAP
Implementujte službu .NET Framework 1 nebo 2 do transportu produktu WebSphere MQ pro SOAP. Vytvořte adresář implementace, spusťte příkaz **amqwdeployWMQService** a spusťte modul listener .NET.

Implementace služby na server CICS Transaction Server pro použití přenosu WebSphere pro protokol SOAP
Přenos produktu WebSphere MQ pro SOAP je integrován do podpory webových služeb CICS Transaction Server 4.1 .

Konfigurace serveru WebSphere Application Server pro použití W3C SOAP přes JMS
Webová služba, která je svázána s doporučujícím doporučením W3C pro protokol SOAP prostřednictvím rozhraní JMS, musí být spuštěna v kontejneru EJB aplikačního serveru Java EE . Tato úloha je krokem 1 připojení klienta webové služby Axis2 a webové služby implementované na serveru WebSphere Application Server pomocí protokolu W3C protokolu SOAP prostřednictvím rozhraní JMS. Nakonfigurujte prostředky WebSphere MQ a WebSphere Application Server pro vývoj a implementaci vazby webových služeb na rozhraní W3C SOAP prostřednictvím rozhraní JMS jako přenosu.

Implementace služby na koncový bod služby WebSphere ESB a Process Server pro použití přenosu WebSphere pro protokol SOAP
Přenos produktu WebSphere MQ pro SOAP není přímo podporován produkty WebSphere ESB a Process Server. Musíte nakonfigurovat vlastní export.

Konfigurace serveru WebSphere Application Server pro použití W3C SOAP přes JMS

Webová služba, která je svázána s doporučujícím doporučením W3C pro protokol SOAP prostřednictvím rozhraní JMS, musí být spuštěna v kontejneru EJB aplikačního serveru Java EE . Tato úloha je krokem 1 připojení klienta webové služby Axis2 a webové služby implementované na serveru WebSphere Application Server pomocí protokolu W3C protokolu SOAP prostřednictvím rozhraní JMS. Nakonfigurujte prostředky WebSphere MQ a WebSphere Application Server pro vývoj a implementaci vazby webových služeb na rozhraní W3C SOAP prostřednictvím rozhraní JMS jako přenosu.

Než začnete

Úloha vyžaduje produkt WebSphere Application Server v7.0.0.9 a WebSphere MQ v7.0.1.3.

Informace o této úloze

Úloha má dva kroky:

Postup

1. “Konfigurace prostředků produktu WebSphere MQ” na stránce 974
2. “Konfigurace prostředků serveru WebSphere Application Server” na stránce 975

Jak pokračovat dále

“Konfigurace prostředků produktu WebSphere MQ” na stránce 974

Související úlohy

Implementace služby na Axis 1.4 k použití pro přenos WebSphere pro protokol SOAP pomocí produktu amqwdeployWMQService

Implementujte službu Axis 1.4 do přenosu WebSphere MQ pro protokol SOAP vytvořením adresáře implementace, spuštěním příkazu **amqwdeployWMQService** a spuštěním modulu listener Axis 1.4 .

Implementace služby pro službu .NET Framework 1 nebo 2 pro použití přenosu WebSphere MQ pro SOAP

Implementujte službu .NET Framework 1 nebo 2 do transportu produktu WebSphere MQ pro SOAP. Vytvořte adresář implementace, spusťte příkaz **amqwdeployWMQService** a spusťte modul listener .NET.

Implementace služby na server CICS Transaction Server pro použití přenosu WebSphere pro protokol SOAP

Přenos produktu WebSphere MQ pro SOAP je integrován do podpory webových služeb CICS Transaction Server 4.1 .

Implementace služby na server WebSphere Application Server pro použití přenosu WebSphere pro protokol SOAP

Přenos protokolu WebSphere MQ pro SOAP je integrován do sběrnice pro integraci služeb na serveru WebSphere Application Server.

Implementace služby na koncový bod služby WebSphere ESB a Process Server pro použití přenosu WebSphere pro protokol SOAP

Přenos produktu WebSphere MQ pro SOAP není přímo podporován produkty WebSphere ESB a Process Server. Musíte nakonfigurovat vlastní export.

Konfigurace prostředků produktu WebSphere MQ

Než začnete

Pro podporu Axis2 je vyžadováno použití produktu WebSphere MQ 7.0.1.3 nebo novější.

Informace o této úloze

Pro zjednodušení úloha předpokládá, že produkt WebSphere MQ je nainstalován na stejné pracovní stanici jako jiný software a používá připojení vazeb. Konfigurace klienta WebSphere Application Server a klienta Axis2 pracuje s připojeními klienta. Chcete-li spustit úlohu pomocí klientských připojení, zkontrolujte, zda lze vkládat zprávy do front požadavků a odpovědí a z nich jak z klienta Axis2 , tak z počítačů s produktem WebSphere Application Server.

Opět platí, že pro zjednodušení není použita žádná konfigurace zabezpečení. ID uživatele má plné oprávnění `mqm` .

Postup

1. Vytvořte výchozího správce front `QM1`.

   Produkt WebSphere MQ Explorer slouží k vytvoření produktu `QM1` jako výchozího správce front. Nakonfigurujte ji tak, aby se spouštěla automaticky, a vyberte volbu pro vytvoření listeneru. Můžete také použít následující příkazy:

   ```
   crtmqm -q -sa QM1
   strmqm
   echo define listener (LISTENER.TCP) trptype(tcp) ipaddr(localhost) port(1414)
              control(qmgr) replace | runmqsc
   echo start listener(LISTENER.TCP) | runmqsc
   ```

2. Definujte frontu požadavků, `REQUESTAXIS`a frontu odpovědí `REPLYAXIS`.

   Použijte Průzkumníka nebo následující příkazy:

   ```
   echo define ql(REQUESTAXIS) replace | runmqsc
   echo define ql(REPLYAXIS) replace | runmqsc
   ```

Jak pokračovat dále

"Konfigurace prostředků serveru WebSphere Application Server" na stránce 975

Konfigurace prostředků serveru WebSphere Application Server

Než začnete

Pro podporu W3C SOAP over JMS vyžadujete produkt WebSphere Application Server v7. Tato konfigurace byla provedena na serveru WebSphere Application Server, verze 7.0 Test Environment v7.0.0.9 Aktualizace 1. Produkt WebSphere Application Server byl dodán s produktem Rational Software Architect for WebSphere Software 7.5.4. Produkt Rational Software Architect byl aktualizován na verzi v7.5.5.1použitím nejnovějších aktualizací, které byly k dispozici.

V rámci instalačního procesu vytvořte profil pro produkt WebSphere Application Server. V úloze není zabezpečení pro administraci povoleno. Výchozí název profilu je `was70profile1`a server je `server1`.

Informace o této úloze

Nakonfigurujte server WebSphere Application Server. Server můžete buď spustit z produktu Rational Application Developer, a spustit administrativní konzolu z pohledu Servery, nebo můžete spustit server pomocí příkazového souboru a spravovat server pomocí prohlížeče. Úloha používá druhou metodu.

Příkazové soubory serveru jsou ve složce, *`Rational Installation Root`*`\SDP\runtimes\base_v7\profiles\was70profile1\bin`. Soubory žurnálu, které byste mohli chtít zkontrolovat, jsou uvedeny v příručce *`Rational Installation Root`*`\SDP\runtimes\base_v7\profiles\was70profile1\logs\server1`.

Jako konvence jsou všechny názvy objektů produktu WebSphere MQ velká a všechny názvy rozhraní JNDI odkazující na objekty produktu WebSphere MQ jsou malými písmeny.

Postup

1. Spusťte server.

   ```
   startServer server1
   ```

2. Spusťte prohlížeč, otevřete konzolu pro správu a přihlaste se.

   ```
   http://localhost:9061/ibm/console/unsecureLogon.jsp
   ```

   Zadejte libovolný řetězec do pole ID uživatele.

3. Vytvořit továrnu na připojení, `qm1`
   a) V navigátoru otevřete **Prostředky** > **JMS** > **Továrny připojení**.
   b) V okně Továrny na připojení vyberte rozsah **Node =*nodename***, klepněte na **New**.
   c) Vyberte volbu **Poskytovatel systému zprávWebSphere MQ** > **OK**.
   d) Poskytněte informace o připojení správce front z produktu Tabulka 143 na stránce 975 > **Další**.

   Tabulka 143. Informace o připojení správce front

Název pole	**Hodnota**
Název	`qm1`
Název rozhraní JNDI	`qm1`

   e) Vyberte volbu **Zadat všechny požadované informace do tohoto průvodce** jako metodu připojení > **Další**.
   f) Zadejte `QM1` jako podrobnosti připojení fronty > **Další**.
   g) Zadejte podrobnosti o připojení z produktu Tabulka 144 na stránce 976 > **Další**.

Tabulka 144. Podrobnosti o připojení	
Název pole	**Hodnota**
Transport	`Bindings, then client`
Název hostitele	`localhost`
Port	`1414`
Kanál připojení serveru	`SYSTEM.DEF.SVRCONN`

h) **Testovat připojení** > **Další** > **Dokončit** > **Uložit**.

4. Vytvořte frontu požadavků JMS, `requestaxis`.
   a) V modulu Navigatorotevřete volbu **Prostředky** > **JMS** > **Fronty**.
   b) V okně Továrny na připojení vyberte rozsah **Node =*nodename***, klepněte na **New**.
   c) Vyberte volbu **Poskytovatel systému zprávWebSphere MQ** > **OK**.
   d) Zadejte podrobnosti o frontě z produktu Tabulka 145 na stránce 976 > **OK** > **Uložit**.

Tabulka 145. Podrobnosti fronty	
Název pole	**Hodnota**
Název	`requestaxis`
Název rozhraní JNDI	`requestaxis`
Název fronty	`REQUESTAXIS`
Název správce front	`QM1`

5. Opakujte krok “4” na stránce 976 a vytvořte frontu odpovědí JMS, `replyaxis`.
6. Vytvořte specifikaci aktivace, `qm1as`.

   Specifikace aktivace spustí objekt MDB (Message Driven Bean) webové služby, když přijde zpráva do fronty požadavků. Objekt MDB je definován v deskriptoru implementace webové služby, která je vytvořena průvodcem webové služby Rational Application Developer.

   a) V modulu Navigatorotevřete volbu **Prostředky** > **JMS** > **Specifikace aktivace**.
   b) V okně Továrny na připojení vyberte rozsah **Node =*nodename***, klepněte na **New**.
   c) Vyberte volbu **Poskytovatel systému zprávWebSphere MQ** > **OK**.
   d) Zadejte základní atributy specifikace aktivace z Tabulka 146 na stránce 976 > **Další**.

Tabulka 146. Název specifikace aktivace	
Název pole	**Hodnota**
Název	`qm1as`
Název rozhraní JNDI	`qm1as`

   e) Zadejte informace o objektu MDB z produktu Tabulka 147 na stránce 976 > **Další**.

Tabulka 147. Informace MDB	
Název pole	**Hodnota**
Název rozhraní JNDI cíle	`requestaxis`
selektor zpráv	*Mezera vlevo*
Typ cíle	`Queue`

   f) Vyberte volbu **Zadat všechny požadované informace do tohoto průvodce** jako metodu připojení > **Další**.

g) Zadejte QM1 jako podrobnosti připojení fronty > **Další**.

h) Zadejte podrobnosti o připojení z produktu Tabulka 144 na stránce 976 > **Další**.

Tabulka 148. Podrobnosti o připojení	
Název pole	**Hodnota**
Transport	`Bindings, then client`
Název hostitele	`localhost`
Port	`1414`
Kanál připojení serveru	`SYSTEM.DEF.SVRCONN`

i) **Testovat připojení** > **Další** > **Dokončit** > **Uložit**.

7. Vytvořte továrnu na připojení fronty `jms/WebServicesReplyQCF`pro frontu odpovědí.

   Směrovač webových služeb používá továrnu připojení fronty pro přístup k frontě odpovědí. V deskriptoru implementace webové služby je pro továrnu připojení fronty poskytnut výchozí název rozhraní JNDI produktu `jms/WebServicesReplyQCF`. Název v deskriptoru implementace můžete změnit. V této úloze přidejte výchozí název do definic prostředků JMS.

   a) V modulu Navigatorotevřete volbu **Prostředky** > **JMS** > **Továrny připojení fronty**.

   b) V okně Továrny na připojení vyberte rozsah **Node =*nodename***, klepněte na **New**.

   c) Vyberte volbu **Poskytovatel systému zprávWebSphere MQ** > **OK**.

   d) Zadejte základní atributy faktorie připojení fronty z produktu Tabulka 149 na stránce 977 > **Další**.

Tabulka 149. Název továrny připojení fronty	
Název pole	**Hodnota**
Název	`WebServicesReplyQCF`
Název rozhraní JNDI	`jms/WebServicesReplyQCF`

   e) Vyberte volbu **Zadat všechny požadované informace do tohoto průvodce** jako metodu připojení > **Další**.

   f) Zadejte QM1 jako podrobnosti připojení fronty > **Další**.

   g) Zadejte podrobnosti o připojení z produktu Tabulka 144 na stránce 976 > **Další**.

Tabulka 150. Podrobnosti o připojení	
Název pole	**Hodnota**
Transport	`Bindings, then client`
Název hostitele	`localhost`
Port	`1414`
Kanál připojení serveru	`SYSTEM.DEF.SVRCONN`

   h) **Testovat připojení** > **Další** > **Dokončit** > **Uložit**.

Jak pokračovat dále

“Vývoj webové služby JAX-WS EJB pro W3C SOAP prostřednictvím rozhraní JMS” na stránce 935

Implementace služby na koncový bod služby WebSphere ESB a Process Server pro použití přenosu WebSphere pro protokol SOAP

Přenos produktu WebSphere MQ pro SOAP není přímo podporován produkty WebSphere ESB a Process Server. Musíte nakonfigurovat vlastní export.

Informace o této úloze

WebSphere Integration Developer provides a SOAP data transformation that you can bind to the WebSphere MQ JMS Export to create a custom WebSphere MQ JMS SOAP Export.

Postupujte podle pokynů a vytvořte upravený export pro příjem požadavků SOAP prostřednictvím přenosu WebSphere MQ pro SOAP.

Postup

1. Přečtěte si téma Přehled importů a exportů a Jak se připojit k produktu WebSphere MQ v dokumentaci k produktu WebSphere Process Server for Multiplatforms V6.2 .
2. Postupujte podle úlohy Generování vazby exportu MQ JMS v dokumentaci produktu IBM Business Process Manager, verze 8.6 .

   Chcete-li formátovat zprávu SOAP, použijte vazbu dat SOAP popsanou v tématu Předbalené transformace formátu dat služby JMS .

Související úlohy

Implementace služby na Axis 1.4 k použití pro přenos WebSphere pro protokol SOAP pomocí produktu amqwdeployWMQService
Implementujte službu Axis 1.4 do přenosu WebSphere MQ pro protokol SOAP vytvořením adresáře implementace, spuštěním příkazu **amqwdeployWMQService** a spuštěním modulu listener Axis 1.4 .

Implementace služby pro službu .NET Framework 1 nebo 2 pro použití přenosu WebSphere MQ pro SOAP
Implementujte službu .NET Framework 1 nebo 2 do transportu produktu WebSphere MQ pro SOAP. Vytvořte adresář implementace, spusťte příkaz **amqwdeployWMQService** a spusťte modul listener .NET.

Implementace služby na server CICS Transaction Server pro použití přenosu WebSphere pro protokol SOAP
Přenos produktu WebSphere MQ pro SOAP je integrován do podpory webových služeb CICS Transaction Server 4.1 .

Implementace služby na server WebSphere Application Server pro použití přenosu WebSphere pro protokol SOAP
Přenos protokolu WebSphere MQ pro SOAP je integrován do sběrnice pro integraci služeb na serveru WebSphere Application Server.

Konfigurace serveru WebSphere Application Server pro použití W3C SOAP přes JMS
Webová služba, která je svázána s doporučujícím doporučením W3C pro protokol SOAP prostřednictvím rozhraní JMS, musí být spuštěna v kontejneru EJB aplikačního serveru Java EE . Tato úloha je krokem 1 připojení klienta webové služby Axis2 a webové služby implementované na serveru WebSphere Application Server pomocí protokolu W3C protokolu SOAP prostřednictvím rozhraní JMS. Nakonfigurujte prostředky WebSphere MQ a WebSphere Application Server pro vývoj a implementaci vazby webových služeb na rozhraní W3C SOAP prostřednictvím rozhraní JMS jako přenosu.

Implementace klientů webových služeb pro použití přenosu produktu WebSphere MQ pro SOAP

Implementujte klienta webové služby na jeden z mnoha různých klientských prostředí a připojte se ke službě pomocí přenosu WebSphere MQ pro SOAP.

Než začnete

Vyvíjejte webovou službu a implementujte ji pro použití přenosu WebSphere MQ pro SOAP.

Informace o této úloze

Klienta webové služby můžete implementovat tak, aby se spouštěl s přenosem WebSphere MQ pro protokol SOAP v řadě různých klientských prostředí. Klienta Java můžete implementovat na Axis 1.4 pouze s použitím softwaru instalovaného s produktem WebSphere MQ. V případě ostatních klientských prostředí je třeba instalovat další software.

Nejste omezeni na spuštění přenosu WebSphere pro protokol SOAP v prostředí klienta, pro které jsou instrukce implementace. Použijte pokyny k implementaci klienta do jednoho z podporovaných prostředí.

Poznámka: Některá integrovaná prostředí nabízejí protokol SOAP prostřednictvím rozhraní JMS s použitím doporučených vazeb JMS SOAP organizace W3C a také přenosu WebSphere MQ pro vazbu SOAP. Verze produktu WebSphere MQ, včetně 7.0.1.2, podporují pouze přenos dat WebSphere MQ pro vazbu SOAP. Počínaje verzí 7.0.1.3 můžete implementovat klienty Axis2 pomocí identifikátoru URI, který odpovídá doporučenému doporučení W3C pro protokol SOAP prostřednictvím rozhraní JMS. Viz výukový program Vývoj aplikací webových služeb SOAP/JMS JAX-WS s produkty WebSphere Application Server V7 a Rational Application Developer V7.5.

Implementace klienta webové služby do Axis 1.4 pro použití přenosu IBM WebSphere MQ pro SOAP

Připravte adresář implementace a deskriptor implementace pro klienta. Zadejte proxy klienta a třídu klienta a nastavte proměnnou prostředí CLASSPATH. Konfigurujte fronty a kanály produktu IBM WebSphere MQ , spusťte službu a otestujte klienta.

Než začnete

Tip: Implementujte službu do protokolu HTTP, vyvinete a otestujte klienta pro protokol HTTP a poté upravte klienta pro transport produktu IBM WebSphere MQ pro SOAP:

1. Přidejte volání Register.extension() do klienta.
2. Změňte adresu statické webové služby ve třídě lokátoru proxy klienta tak, aby používala identifikátor URI pro přenos IBM WebSphere MQ pro protokol SOAP.

Informace o této úloze

Implementace klienta Axis 1.4 pro použití přenosu IBM WebSphere MQ pro SOAP vyžaduje jeden další krok implementace ve srovnání s klientem HTTP. Chcete-li mapovat transport `jms:` na třídu odesílatele `com.ibm.mq.soap.transport.jms.WMQSender`, musíte vytvořit deskriptor implementace klienta `client-config.wsdd`.

Pokud používáte příkaz **amqwdeployWMQService** ke generování klientských serverů proxy, můžete implementovat klienta pomocí adresářů, které příkaz vygeneruje.

Postup

1. Vytvořte adresář *deployDir* , kde budou uloženy soubory implementace klienta.
2. Otevřete příkazové okno v systémech Windows nebo příkazový shell pomocí systému X Window System na systémech UNIX and Linux v adresáři *deployDir*.
3. Spuštěním příkazu **amqwsetcp.cmd** nastavte proměnnou prostředí CLASSPATH .
4. Spuštěním příkazu **amqwclientconfig.cmd** vytvořte deskriptor implementace klienta Axis 1.4 , `client-config.wsdd` v adresáři *deployDir*.
5. Ujistěte se, že třídy v balíku klienta, třídy proxy klienta a knihovny, které klient používá, jsou uvedeny v proměnné CLASSPATH.

   Produkt **amqwdeployWMQService** umístí proxy klienta .NET do `./generated/server/soap/client/remote/dotnetService` a proxy Axis 1.4 do `./generated/server/soap/client/remote/`*`client package`*.

Příklad

Příklad ukazuje konfiguraci a výstup, Obrázek 199 na stránce 980, z klienta Axis 1.4 Java . Klient, Obrázek 198 na stránce 980, volá webovou službu, která odráží jeho vstupní parametr. Definice služby, Obrázek 197 na stránce 980, zobrazuje identifikátor URI převzaté ze souboru WSDL služby.

```
<wsdl:service name="QuoteSOAPImplService">
    wsdl:port binding="intf:org.example.www.QuoteSOAPImplBindingSoap"
                      name="org.example.www.QuoteSOAPImpl_Wmq">
       <wsdlsoap:address location="jms:/queue?destination=REQUESTAXIS
           &connectionFactory=(connectQueueManager(QM1)binding(server))
           &initialContextFactory=com.ibm.mq.jms.Nojndi
           &targetService=org.example.www.QuoteSOAPImpl.java
           &replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE" />
    </wsdl:port>
</wsdl:service>
```

Obrázek 197. definice služby

```
package org.example.www;
import com.ibm.mq.soap.Register;
public class QuoteClient {
   public static void main(String[] args) {
      try {
         Register.extension();
         QuoteSOAPImplServiceLocator locator = new QuoteSOAPImplServiceLocator();
         System.out.println("Response = "
                   + locator.getOrgExampleWwwQuoteSOAPImpl_Wmq().getQuote("IBM"));
      } catch (Exception e) {
         System.out.println("Exception = " + e.getMessage());
      }
   }
}
```

Obrázek 198. Klient Axis 1.4 Java

```
C:\IBM\ID\Test>dir /s /b
C:\IBM\ID\Test\client-config.wsdd
C:\IBM\ID\Test\org
C:\IBM\ID\Test\org\example
C:\IBM\ID\Test\org\example\www
C:\IBM\ID\Test\org\example\www\GetQuoteFaultMsg.class
C:\IBM\ID\Test\org\example\www\OrgExampleWwwQuoteSOAPImplBindingSoapStub.class
C:\IBM\ID\Test\org\example\www\QuoteClient.class
C:\IBM\ID\Test\org\example\www\QuoteSOAPImpl.class
C:\IBM\ID\Test\org\example\www\QuoteSOAPImplService.class
C:\IBM\ID\Test\org\example\www\QuoteSOAPImplServiceLocator.class

C:\IBM\ID\Test>amqwsetcp
C:\IBM\ID\Test>java org.example.www.QuoteClient.class
Response = IBM
```

Obrázek 199. Konfigurace a výstup klienta

Jak pokračovat dále

1. Implementujete-li klienta jako klienta IBM WebSphere MQ , nakonfigurujte kanál připojení klienta a serveru.
2. Implementujete-li klienta do jiného správce front do služby, je třeba zpřístupnit cílovou frontu klientovi. Konfigurujte cílovou frontu ve správci front služeb jako frontu klastru nebo ve správci front klienta jako definici vzdálené fronty.

Související úlohy

Implementace klienta webové služby do prostředí Axis2 pro použití přenosu WebSphere MQ pro SOAP
Připravte adresář implementace a konfigurační soubor Axis2 pro klienta. Zadejte proxy klienta a třídu klienta a nastavte proměnnou CLASSPATH. Nakonfigurujte fronty a kanály produktu WebSphere MQ , spusťte danou službu a otestujte klienta.

Implementace na klienta Axis2 pomocí protokolu W3C SOAP prostřednictvím rozhraní JMS
Webová služba, která je svázána s doporučujícím doporučením W3C pro protokol SOAP prostřednictvím rozhraní JMS, musí být spuštěna v kontejneru EJB aplikačního serveru Java EE . Tato úloha je krokem

4 připojení klienta webové služby Axis2 a webové služby implementované na serveru WebSphere Application Server pomocí protokolu W3C protokolu SOAP prostřednictvím rozhraní JMS. Upravte adresu URL v klientovi Axis2 vyvinutém pro přenos WebSphere MQ pro protokol SOAP tak, aby bylo možné použít doporučení kandidáta W3C pro protokol SOAP prostřednictvím rozhraní JMS.

Implementace klienta webové služby do prostředí .NET Framework 1 a 2 pro použití přenosu WebSphere MQ pro SOAP
Připravte adresář implementace a deskriptor implementace pro klienta. Zadejte proxy klienta a třídu klienta. Nakonfigurujte fronty a kanály produktu WebSphere MQ , spusťte danou službu a otestujte klienta.

Implementace klienta webové služby do prostředí Axis2 pro použití přenosu WebSphere MQ pro SOAP

Připravte adresář implementace a konfigurační soubor Axis2 pro klienta. Zadejte proxy klienta a třídu klienta a nastavte proměnnou CLASSPATH. Nakonfigurujte fronty a kanály produktu WebSphere MQ , spusťte danou službu a otestujte klienta.

Než začnete

Tip: Implementujte službu do protokolu HTTP. Vyvíjejte a otestujte klienta pro protokol HTTP a poté upravte adresu URL tak, aby odkazovaly na službu pomocí přenosu protokolu SOAP produktu WebSphere MQ .

Tato úloha ukazuje, jak implementovat nespravovaný klient Axis2 do prostředí Java Standard Edition. Je možné, že budete chtít implementovat klienta Axis2 do webového kontejneru. V produktu “Vyvíjení klienta JAX-WS pro přenos WebSphere pro protokol SOAP pomocí prostředí Eclipse” na stránce 949jste vyvinuli klienta ve webovém kontejneru a implementovali jste ho na server WebSphere Application Server Community Edition. Jako součást konfigurace serveru jste povolili fasetu Axis2 a zahrnuli jste fasetu v konfiguraci webového kontejneru. Chcete-li nakonfigurovat webové kontejnery na jiných aplikačních serverech, přečtěte si dokumentaci Axis2 , http://ws.apache.org/axis2/1_4_1/installationguide.html#servlet_containernebo dokumentaci dodanou s webovým serverem.

Poznámka: Axis2 používá termín, kontejner servletu. Kontejner servletů je stejný jako webový kontejner.

Informace o této úloze

Implementace klienta Axis2 pro použití přenosu WebSphere MQ pro SOAP je jako implementace klienta Axis2 pro použití protokolu HTTP. Další kroky jsou vyžadovány k poskytnutí cesty ke třídám pro soubory JAR produktu WebSphere MQ a k úpravě konfiguračního souboru Axis2 . Konfigurační soubor Axis2 vyžaduje další položku pro službu JMS. Tato položka odkazuje na přenos WebSphere MQ pro soubor JAR SOAP, který implementuje rozhraní JMS `transportSender`.

Axis2 poskytuje skript, `axis2.bat` nebo `axis2.sh`, který zjednodušuje implementaci klienta; viz příklady v Obrázek 203 na stránce 983 a Obrázek 204 na stránce 984.

Poznámka:

1. `axis2.bat` má chybu, která musí být opravena. Řetězec `-Djava.ext.dirs="%AXIS2_HOME%\lib\"` musí být změněn na `-Djava.ext.dirs="%AXIS2_HOME%\lib\\"`.
2. V produktu `axis2.bat` a `axis2.sh`se produkt `-Djava.ext.dirs` používá jako rychlý způsob, jak odkazovat na všechny soubory JAR Axis2 místo jejich přidání do cesty ke třídě. Tento přístup je bohužel chybný a funguje pouze s některými prostředími JRE. Nepracuje s prostředím JRE IBM .

   Parametr prostředí JVM `-Djava.ext.dirs="%AXIS2_HOME%\lib\\"`zpřístupňuje soubory JAR Axis pro prostředí JVM. Prostředí JVM se pokouší o vytvoření instance některých souborů JAR Axis a vede k chybě, jejíž podrobnosti závisejí na prostředí JVM. Obvykle se v trasování zásobníku může zobrazit jeden z následujících řádků:

   ```
   org.apache.axiom.om.util.UUIDGenerator.getInitialUUID(UUIDGenerator.java:76)
   ```

   , nebo `org.apache.axis2.deployment.DeploymentException: java.security.NoSuchAlgorithmException: MD5 MessageDigest not available`

Správným způsobem spuštění nespravovaného klienta Axis2 je přidání souborů JAR Axis2 do cesty ke třídám. Cesta ke třídám je k dispozici pouze pro aplikaci klienta a nikoli pro prostředí JVM.

Procedura popisuje obecné kroky ke spuštění nespravovaného klienta Axis2 bez použití skriptu axis2 . Příklady v Obrázek 201 na stránce 983 a Obrázek 202 na stránce 983 jsou skripty pro Windows a Linux.

Postup

1. Stáhněte si Axis2 1.4.1 z webu http://ws.apache.org/axis2/download/1_4_1/download.cgi a rozbalte ji do složky `Axis2-1.4.1`.
2. Aktualizujte produkt `axis2.xml` v produktu `Axis2-1.4.1\conf`.
   a) Aktualizujte produkt `axis2.xml` v produktu `Axis2-1.4.1\conf`. Přidejte transport produktu WebSphere MQ pro SOAP jako `transportSender`:

   ```
   <transportSender name="jms"
   class="com.ibm.mq.axis2.transport.jms.WMQJMSTransportSender"/>
   ```

   b) V případě potřeby změňte velikost fondu připojení z výchozího nastavení 10.

   ```
   <transportSender name="jms"
   class="com.ibm.mq.axis2.transport.jms.WMQJMSTransportSender">
   <parameter name="ResourcePoolCapacity">20</parameter>
   </transportSender>
   ```

   Položka `ResourcePoolCapacity` definuje, kolik položek koncového bodu služby je uchováno v mezipaměti. Hodnota musí být alespoň 1. Pokud počet položek koncového bodu služby překročí velikost mezipaměti, položky se odstraní, aby se místo nových záznamů neslo o místo. Velikost záznamu koncového bodu se mění. Nastavte číslo, které je dostatečně velké, aby se zabránilo hluchému stránkování mezipaměti.

   Viz krok 3 v tématu “Vyvíjení klienta JAX-WS pro přenos WebSphere pro protokol SOAP pomocí prostředí Eclipse” na stránce 949.
3. Vytvořte adresář *deployDir*. Pod tímto adresářem zkopírujte strukturu složek obsahující proxy klienta a klienta. *deployDir* je ekvivalent ke složce *`project`*`\bin` v projektu Java Eclipse .
4. Otevřete příkazové okno na systému Windowsnebo příkazový shell pomocí systému X Window System na systémech UNIX and Linux v adresáři *deployDir*.
5. Aktualizujte cestu ke třídě, aby zahrnovala aktuální adresář, Axis2 soubory JAR, `com.ibm.mqjms.jar` a `com.ibm.mq.axis2.jar`.

   `com.ibm.mqjms.jar` se odkazuje na všechny ostatní soubory JAR produktu WebSphere MQ , které jsou povinné.
6. Chcete-li spustit klientský program, použijte příkaz **Java** .

Příklady

Čtyři příklady spuštění klienta Axis2 jsou uvedeny v dokumentu Obrázek 202 na stránce 983 produktu Obrázek 204 na stránce 984. Obrázek 200 na stránce 983 zobrazuje výstup od spuštění asynchronního klienta uvedeného v části Obrázek 183 na stránce 954.

```
cd C:\IBM\ID\Workspaces\Axis2docs\StockQuoteAxis2PojoClient\bin>
runpojo soap/client/SQA2AsyncClient

HTTP Sync: 55.25
Callback constructor
Waiting for HTTP callback
Result in Callback 55.25
HTTP poll: 55.25
JMS: Sync: 55.25
Callback constructor
Waiting for JMS callback
Result in Callback 55.25
JMS poll: 55.25
Press any key to continue . . .
```

Obrázek 200. Výstup ze spuštění SQA2AsyncClient

```
@echo off
set AXIS2_HOME=C:\OpenSource\axis2-1.4.1
set JAVA_HOME=C:\IBM\Java50
set WMQ_HOME=C:\IBM\MQ\java\lib

setlocal EnableDelayedExpansion
set CLASSPATH=
set AXIS2_CLASS_PATH=
FOR %%c in ("%AXIS2_HOME%\lib\*.jar") DO set AXIS2_CLASS_PATH=!AXIS2_CLASS_PATH!;%%c

"%JAVA_HOME%\bin\java" -Daxis2.repo="%AXIS2_HOME%\repository"
-Daxis2.xml="%AXIS2_HOME%\conf\axis2.xml" -cp
".;%WMQ_HOME%\com.ibm.mqjms.jar;%WMQ_HOME%\com.ibm.mq.axis2.jar;%AXIS2_CLASS_PATH%" %1

pause
```

Obrázek 201. runpojo.bat: Windowspomocí cesty ke třídě

```
export AXIS2_HOME=/home/OpenSource/axis2-1.4.1
export JAVA_HOME=/usr/lib/j2sdk1.5-ibm
# update classpath
AXIS2_CLASSPATH=""
for f in "$AXIS2_HOME"/lib/*.jar
do
  AXIS2_CLASSPATH="$AXIS2_CLASSPATH":$f
done
AXIS2_CLASSPATH="$AXIS2_HOME":"$JAVA_HOME/lib/tools.jar":"$AXIS2_CLASSPATH":"$CLASSPATH"
java -cp /home/alex/dev/sandbox/Soap/axis2:/opt/mqm/java/lib/com.ibm.mqjms.jar:
/opt/mqm/java/lib/com.ibm.mq.axis2.jar:$AXIS2_CLASSPATH
-Daxis2.xml=/home/alex/dev/sandbox/axis2-1.4.1/conf/axis2.xml %1
```

Obrázek 202. runpojo.sh: Linuxpomocí cesty ke třídě.

```
@echo off
set AXIS2_HOME=C:\OpenSource\axis2-1.4.1
set JAVA_HOME=C:\IBM\Java50
set WMQ_HOME=C:\IBM\MQ\java\lib

%AXIS2_HOME%\bin\axis2 -cp .;%WMQ_HOME%\com.ibm.mqjms.jar;%WMQ_HOME%\com.ibm.mq.axis2.jar; %1
pause
```

Obrázek 203. runaxis2.bat: Windows, pomocí axis2.bat

Poznámka

```
export AXIS2_HOME=/home/OpenSource/axis2-1.4.1
export JAVA_HOME=/usr/lib/j2sdk1.5-ibm

%AXIS2_HOME%\bin\axis2 -cp .;%WMQ_HOME%\com.ibm.mqjms.jar;%WMQ_HOME%\com.ibm.mq.axis2.jar; %1
```

Obrázek 204. runaxis2.sh: Linux, použití axis2.sh

Poznámka

Související úlohy

Implementace klienta webové služby do Axis 1.4 pro použití přenosu IBM WebSphere MQ pro SOAP
Připravte adresář implementace a deskriptor implementace pro klienta. Zadejte proxy klienta a třídu klienta a nastavte proměnnou prostředí CLASSPATH. Konfigurujte fronty a kanály produktu IBM WebSphere MQ , spusťte službu a otestujte klienta.

Implementace na klienta Axis2 pomocí protokolu W3C SOAP prostřednictvím rozhraní JMS
Webová služba, která je svázána s doporučujícím doporučením W3C pro protokol SOAP prostřednictvím rozhraní JMS, musí být spuštěna v kontejneru EJB aplikačního serveru Java EE . Tato úloha je krokem 4 připojení klienta webové služby Axis2 a webové služby implementované na serveru WebSphere Application Server pomocí protokolu W3C protokolu SOAP prostřednictvím rozhraní JMS. Upravte adresu URL v klientovi Axis2 vyvinutém pro přenos WebSphere MQ pro protokol SOAP tak, aby bylo možné použít doporučení kandidáta W3C pro protokol SOAP prostřednictvím rozhraní JMS.

Implementace klienta webové služby do prostředí .NET Framework 1 a 2 pro použití přenosu WebSphere MQ pro SOAP
Připravte adresář implementace a deskriptor implementace pro klienta. Zadejte proxy klienta a třídu klienta. Nakonfigurujte fronty a kanály produktu WebSphere MQ , spusťte danou službu a otestujte klienta.

Implementace na klienta Axis2 pomocí protokolu W3C SOAP prostřednictvím rozhraní JMS

Webová služba, která je svázána s doporučujícím doporučením W3C pro protokol SOAP prostřednictvím rozhraní JMS, musí být spuštěna v kontejneru EJB aplikačního serveru Java EE . Tato úloha je krokem 4 připojení klienta webové služby Axis2 a webové služby implementované na serveru WebSphere Application Server pomocí protokolu W3C protokolu SOAP prostřednictvím rozhraní JMS. Upravte adresu URL v klientovi Axis2 vyvinutém pro přenos WebSphere MQ pro protokol SOAP tak, aby bylo možné použít doporučení kandidáta W3C pro protokol SOAP prostřednictvím rozhraní JMS.

Než začnete

Nejprve musíte dokončit úlohu, “Vyvíjení klienta JAX-WS pro přenos WebSphere pro protokol SOAP pomocí prostředí Eclipse” na stránce 949 pro volání **SimpleJavaListener** pomocí klienta Axis2 a přenosu WebSphere MQ pro protokol SOAP.

Musíte také vytvořit webovou službu a nakonfigurovat produkt WebSphere MQ a server WebSphere Application Server v předchozích úlohách:

1. “Konfigurace prostředků produktu WebSphere MQ” na stránce 974.
2. “Konfigurace prostředků serveru WebSphere Application Server” na stránce 975.
3. “Vývoj webové služby JAX-WS EJB pro W3C SOAP prostřednictvím rozhraní JMS” na stránce 935.

V rámci úlohy je klient spuštěn v systému Eclipse Galileo. Klienta můžete spustit z příkazového řádku tak, že upravíte soubor `Axis2.bat` dodávaný s Axis2.

Informace o této úloze

Jediná změna, kterou musíte provést na stávajícím statickém klientovi Axis2 `StockQuoteAxis` pro volání služby Axis StockQuotehostované serverem WebSphere Application Server, je změna adresy URL, která byla předána klientovi. Vzhledem k tomu, že se WSDL nezměnil, můžete použít stejné třídy proxy v balíku `soap.server`.

K definování adresy URL, která má být předána klientovi, máte dva přístupy. Můžete použít stejnou adresu URL jako v generovaném souboru `StockQuoteAxis.wsdl`. Musíte přidat parametry `jndiInitialContextFactory` a `jndiURL` pro přístup k adresáři WebSphere Application Server JNDI. Jiným přístupem je změna adresy URL a poskytnutí přímého přístupu klienta k frontám `REQUESTAXIS` a `REPLYAXIS` na serveru `QM1`bez použití vyhledávání JNDI.

Parametry připojení, které definujete v adrese URL předané klientovi Axis2 , se používají pro připojení ke správci front WebSphere MQ a k frontám, které jsou vyžadovány k odeslání a přijetí zpráv SOAP. Parametry připojení předané klientovi Axis2 nemusí být nutně používány službou. Pomocí možností distribuovaných front produktu WebSphere MQ můžete odpojit klienta a službu od použití stejného správce front nebo stejného serveru názvů.

Postup

1. Uložte adresu URL z generované `StockQuoteAxis.wsdl` a zavřete produkt Rational Application Developer, abyste jej uložili do paměti.

   Pokud jste nezměnili konfiguraci serveru, zavřením produktu Rational Application Developer se aplikační server zastaví. V takovém případě spusťte server s příkazem:

   ```
   startserver server1
   ```

2. Otevřete projekt Eclipse Galileo v pracovním prostoru s klientským projektem Axis2 .
3. Otevřete produkt `SQA2StaticClient.java`.

   Viz SQA2StaticClient.java.
4. Volejte službu pomocí varianty `queue` s použitím varianty identifikátoru URI.
   a) Upravte adresu URL.

   Nový URI je:

   ```
   jms:queue:REQUESTAXIS
             ?replyToName=REPLYAXIS
             &connectionFactory=connectQueueManager(QM1)Bind(Server)
             &targetService=StockQuoteAxis;
   ```

   Porovnejte je s adresou URL z produktu `StockQuoteAxis.wsdl`:

   ```
   jms:jndi:requestaxis
            ?jndiConnectionFactoryName=qm1
            &targetService=StockQuoteAxis
   ```

   Obrázek 205. Adresa URL od `StockQuoteAxis.wsdl`

   - `REQUESTAXIS` je nyní velká písmena, protože jde o název fronty a nikoli na název rozhraní JNDI.
   - Připojení k produktu `QM1` je jednoduché.
   - Identifikátor URI neobsahuje název odpovědi na místo určení. Klient musí definovat frontu, na které očekává odpovědi.

   b) Spusťte produkt `SQA2StaticClient.java` s použitím stejného příkazu **Spustit jako ...** konfigurace, jak jste provedli v úloze “Vyvíjení klienta JAX-WS pro přenos WebSphere pro protokol SOAP pomocí prostředí Eclipse” na stránce 949.
5. Volejte službu pomocí varianty identifikátoru URI produktu `jndi` pomocí serveru WebSphere Application Server jako serveru názvů.
   a) Použijte adresu URL z produktu `StockQuoteAxis.wsdl`, Obrázek 205 na stránce 985, poskytující chybějící parametry pro použití služby názvů na serveru WebSphere Application Server.

   Chybějící parametry a hodnoty, které musíte zadat, jsou:

Tabulka 151. Další parametry JNDI		
Parametr	**Hodnota použitá v tomto příkladu**	**Popis**
`&jndiURL`	`iiop://localhost:2810`, nebo `corbaname:iiop:localhost:2810`	Identifikátor URI poskytovatele názvů. Pro produkt WebSphere Application Server je výchozí hodnota nastavena na 2809. Je také znám jako číslo portu konektoru RMI a port samozavedení. Hodnota je uvedena v seznamu `SystemOut.log` `00000000 NameServerImp A NMSV0018I: Name server available on bootstrap port 2810`
`&jndiInitialContextFactory`	`com.ibm.websphere.naming.WsnInitialContextFactory`	Název počáteční kontextové továrny používané produktem WebSphere Application Server.
`&replyToName`	`replyaxis`	Název rozhraní JNDI fronty REPLYAXIS .

```
jms:jndi:requestaxis?
         &jndiURL=iiop//localhost:2810
         &jndiConnectionFactoryName=qm1
         &jndiInitialContextFactory=com.ibm.websphere.naming.WsnInitialContextFactory
         &targetService=StockQuoteAxis
         &replyToName=replyaxis;
```

b) Přidejte soubory JAR požadované vyhledáváním v rozhraní JNDI.

V této konfiguraci byly do cesty sestavení přidány následující soubory JAR, aby bylo možné úlohu spustit s použitím varianty `jndi` adresy URL služby JMS:

- `com.ibm.jaxws.thinclient_7.0.0.jar` z *`Rational install directory`*`\SDP\runtimes\base_v7\runtimes`.
- `com.ibm.ws.runtime.jar` z *`Rational install directory`*`\SDP\runtimes\base_v7\plugins`

Pro jiného poskytovatele rozhraní JNDI je třeba použít různé soubory JAR.

Další soubory JAR v cestě sestavení jsou:

i) Všechny soubory JAR v produktu *`WebSphere MQ Install directory`*`\java\lib`.

ii) Všechny soubory JAR v produktu *`Axis2-1.5.1`*`\lib`.

iii) Prostředí JRE Java 6.0 .

c) Spusťte produkt `SQA2StaticClient.java` s použitím stejného příkazu **Spustit jako ...** konfigurace, jak jste provedli v úloze “Vyvíjení klienta JAX-WS pro přenos WebSphere pro protokol SOAP pomocí prostředí Eclipse” na stránce 949.

Výsledky

V obou případech se odpověď ze služby zobrazí v pohledu konzoly klienta.

Související úlohy

Implementace klienta webové služby do Axis 1.4 pro použití přenosu IBM WebSphere MQ pro SOAP

Připravte adresář implementace a deskriptor implementace pro klienta. Zadejte proxy klienta a třídu klienta a nastavte proměnnou prostředí CLASSPATH. Konfigurujte fronty a kanály produktu IBM WebSphere MQ , spusťte službu a otestujte klienta.

Implementace klienta webové služby do prostředí Axis2 pro použití přenosu WebSphere MQ pro SOAP
Připravte adresář implementace a konfigurační soubor Axis2 pro klienta. Zadejte proxy klienta a třídu klienta a nastavte proměnnou CLASSPATH. Nakonfigurujte fronty a kanály produktu WebSphere MQ , spusťte danou službu a otestujte klienta.

Implementace klienta webové služby do prostředí .NET Framework 1 a 2 pro použití přenosu WebSphere MQ pro SOAP
Připravte adresář implementace a deskriptor implementace pro klienta. Zadejte proxy klienta a třídu klienta. Nakonfigurujte fronty a kanály produktu WebSphere MQ , spusťte danou službu a otestujte klienta.

Implementace klienta webové služby do prostředí .NET Framework 1 a 2 pro použití přenosu WebSphere MQ pro SOAP

Připravte adresář implementace a deskriptor implementace pro klienta. Zadejte proxy klienta a třídu klienta. Nakonfigurujte fronty a kanály produktu WebSphere MQ , spusťte danou službu a otestujte klienta.

Než začnete

Tip: Vytvořte a otestujte službu a klienta pomocí produktu Visual Studio. Poté upravte klienta pro přenos WebSphere MQ pro SOAP.

1. Implementujete-li službu pomocí rozhraní .NET Framework 1 nebo 2, sestavte službu jako knihovnu (`.dll`). Implementace pomocí přenosu WebSphere MQ pro SOAP.
2. Přidejte volání Register.Extension() do klienta.
3. Přidejte odkaz na soubor amqsoap.dll, který se nachází v produktu *`MQ_Install`*`\bin`.
4. Změňte statickou vlastnost `Url` v konstruktoru třídy proxy klienta na identifikátor URI produktu `jms:/` , pro přenos WebSphere MQ pro SOAP.

Informace o této úloze

Implementace klienta webové služby pro platformu .NET Framework 1 nebo 2 k použití přenosu WebSphere MQ pro SOAP vyžaduje další krok implementace. Musíte zaregistrovat produkt `amqsoap.dll` s produktem .NET Framework. Produkt `amqsoap.dll` je automaticky registrován jako součást instalace produktu WebSphere MQ pro protokol SOAP, ale možná jej budete muset registrovat znovu.

Pokud používáte příkaz **amqwdeployWMQService** ke generování klientských serverů proxy, můžete implementovat klienta pomocí adresářů, které příkaz vygeneruje.

Postup

1. Vytvořte adresář *deployDir* , kde budou uloženy soubory implementace klienta.
2. Otevřete příkazové okno v adresáři *deployDir*.
3. Spuštěním příkazu **amqwsetcp** nastavte proměnnou prostředí CLASSPATH , pokud má být služba spuštěna na ose Axis 1.4.
4. V případě potřeby spusťte **amqwRegisterDotNet** pro registraci `amqsoap.dll` s platformou .NET Framework.

Příklad

Příklad ukazuje konfiguraci a výstup, Obrázek 208 na stránce 988, z klienta .NET Framework V2 . Klient, Obrázek 207 na stránce 988, volá webovou službu, která odráží svůj vstupní parametr. Statická definice URL, Obrázek 206 na stránce 988, ukazuje konstruktor pro proxy klienta.

```
public Quote() {
    this.Url = "jms:/queue?destination=REQUESTDOTNET
               &connectionFactory=(connectQueueManager(QM1)binding(server))
               &initialContextFactory=com.ibm.mq.jms.Nojndi
               &targetService=Quote.asmx
               &replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE";
}
```

Obrázek 206. Statický konstruktor klienta proxy

```
using System;
namespace QuoteClientProgram {
    class QuoteMain {
        static void Main(string[] args) {
            try {
                IBM.WMQSOAP.Register.Extension();
                Quote q = new Quote();
                Console.WriteLine("Response is: " + q.getQuote("ibm"));
            } catch (Exception e) {
                Console.WriteLine("Exception is: " + e);
            }
        }
    }
}
```

Obrázek 207. Klientský program

```
C:\IBM\ID\DotNet\QuoteClientProgram\QuoteClientProgram>dir /s /b
C:\IBM\ID\DotNet\QuoteClientProgram\QuoteClientProgram\QuoteClientProgram.exe
C:\IBM\ID\DotNet\QuoteClientProgram\QuoteClientProgram>quoteclientprogram
Response is: IBM
```

Obrázek 208. Konfigurace a výstup

Jak pokračovat dále

1. Implementujete-li klienta jako klienta WebSphere MQ MQI, nakonfigurujte kanál připojení klienta a serveru.
2. Implementujete-li klienta do jiného správce front do služby, je třeba zpřístupnit cílovou frontu klientovi. Konfigurujte cílovou frontu ve správci front služeb jako frontu klastru nebo ve správci front klienta jako definici vzdálené fronty.

Související úlohy

Implementace klienta webové služby do Axis 1.4 pro použití přenosu IBM WebSphere MQ pro SOAP
Připravte adresář implementace a deskriptor implementace pro klienta. Zadejte proxy klienta a třídu klienta a nastavte proměnnou prostředí CLASSPATH. Konfigurujte fronty a kanály produktu IBM WebSphere MQ , spusťte službu a otestujte klienta.

Implementace klienta webové služby do prostředí Axis2 pro použití přenosu WebSphere MQ pro SOAP
Připravte adresář implementace a konfigurační soubor Axis2 pro klienta. Zadejte proxy klienta a třídu klienta a nastavte proměnnou CLASSPATH. Nakonfigurujte fronty a kanály produktu WebSphere MQ , spusťte danou službu a otestujte klienta.

Implementace na klienta Axis2 pomocí protokolu W3C SOAP prostřednictvím rozhraní JMS
Webová služba, která je svázána s doporučujícím doporučením W3C pro protokol SOAP prostřednictvím rozhraní JMS, musí být spuštěna v kontejneru EJB aplikačního serveru Java EE . Tato úloha je krokem 4 připojení klienta webové služby Axis2 a webové služby implementované na serveru WebSphere Application Server pomocí protokolu W3C protokolu SOAP prostřednictvím rozhraní JMS. Upravte adresu URL v klientovi Axis2 vyvinutém pro přenos WebSphere MQ pro protokol SOAP tak, aby bylo možné použít doporučení kandidáta W3C pro protokol SOAP prostřednictvím rozhraní JMS.

Připojte klienta Axis2 k službě rozhraní JAX-WS pomocí protokolu W3C SOAP prostřednictvím rozhraní JMS a serveru WebSphere Application Server.

Po dokončení této úlohy se bude volat webová služba JAX-WS běžící na aplikačním serveru WebSphere z klienta Axis2 . Klient Axis2 a server WebSphere Application Server používají doporučení kandidáta W3C pro protokol SOAP prostřednictvím rozhraní JMS spouštěný v produktu WebSphere MQ. Pomocí produktů Eclipse Galileo a Rational Application Developer můžete sestavit klienta webové služby a webovou službu.

Než začnete

Úloha vyžaduje verzi 7 produktu Rational Software Development Environment a serveru WebSphere Application Server. Úloha byla vytvořena s použitím produktu Rational Application Developer zabaleného s produktem Rational Software Architect for WebSphere Software v7.5.5.1a WebSphere Application Server verze 7.0 Test Environment v7.0.0.9 Aktualizace 1. Také vyžadujete produkt WebSphere MQ v7.0.1.3.

Tato úloha je založena na dvou dalších úlohách, “Vývoj služby JAX-RPC pro přenos WebSphere MQ pro SOAP pomocí prostředí Eclipse” na stránce 928a “Vyvíjení klienta JAX-WS pro přenos WebSphere pro protokol SOAP pomocí prostředí Eclipse” na stránce 949. Chcete-li dokončit tyto úlohy, vaše vývojové prostředí již má nainstalovaný produkt Eclipse Galileo, WASCE, modul plug-in Eclipse pro WASCE a Axis2 1.4.1 . Pro tuto úlohu není vyžadováno WASCE.

Některé kroky jsou složité. Tento postup předpokládá, že při vyvíjení aplikací webových služeb pro produkt WebSphere Application Server pomocí nástroje Rational Application Developer bude určitá znalost. Požadavky na procesor a paměť pro úlohu jsou velké. Úloha byla provedena ve virtuálním počítači VMWare Windows XP SP3 přiděleném 1.8GB paměti.

Nainstalujte všechen software před spuštěním úlohy. Software trvá asi den stažení a jeden den k instalaci, v závislosti na šířce pásma. Úloha trvá alespoň půl dne.

Informace o této úloze

Scénář pro tuto úlohu spočívá v tom, že jste vyvinuli webovou službu s cenovou nabídkou, `StockQuoteAxis`pomocí otevřeného zdrojového nástroje Eclipse Galileo. Produkt `StockQuoteAxis` je implementován pomocí protokolu SOAP prostřednictvím protokolu HTTP, který je spuštěn na otevřeném zdrojovém serveru WASCE.

Chcete svázat webové služby, které implementujete na přenos systému zpráv založené na standardu, například SOAP prostřednictvím rozhraní JMS, nebo na spolehlivý systém zpráv webových služeb a také na protokol SOAP prostřednictvím protokolu HTTP. Chcete, aby klient i služba používaly standardní rozhraní založená na rozhraní. Z tohoto důvodu, ačkoli vývojový tým vašich budoucích projektů implementoval řešení pomocí přenosu WebSphere MQ pro SOAP, jste se nevrátili do výroby.

Klient Axis2 odstranil problém, který klient SOAP pro přenos WebSphere MQ pro protokol SOAP vyžaduje změnu z klienta HTTP. Problém zůstal i nadále, že služba připojená přenosem IBM WebSphere MQ pro SOAP je hostována speciálním modulem listener poskytovaným produktem WebSphere MQ: `SimpleJavaListener`.

Při použití standardu W3C SOAP přes standard JMS ve stavu doporučeného doporučení někteří dodavatelé poskytují podporu pro protokol W3C SOAP prostřednictvím rozhraní JMS. Podpora vám umožňuje implementovat webovou službu na aplikační server a připojit se ke stejné službě pomocí různých protokolů konektivity. Podpora poskytovaná serverem WebSphere Application Server v7 odstraňuje problém s tím, že je nutné hostovat webovou službu samostatně, aby bylo možné použít přenos SOAP založený na zprávách. Použití rozhraní pro přenos zpráv s rozhraním JMS, znamená to, že můžete vyvíjet řešení pomocí nástrojů různých dodavatelů. Doufáme, že nástroje webových služeb na platformě Eclipse budou v budoucnu zahrnovat vazby SOAP přes JMS.

Většina kroků se provádí pomocí platformy Eclipsenebo pomocí nástrojů správy dodávaných s produkty WebSphere . Tyto kroky jsou popsány v prostředí Windows . S drobnými úpravami některých příkazů můžete provádět tyto kroky na jiných platformách.

Vypíší se přípravné kroky pro vytvoření webové služby HTTP a připojení k ní pomocí prostředí Axis2 . Klient a WSDL z těchto kroků se používají k vytvoření řešení.

Postup

1. Připojte se k webové službě Axis StockQuotepomocí klienta Axis2 a přenosu IBM WebSphere MQ pro SOAP
   a) “Vývoj služby JAX-RPC pro přenos WebSphere MQ pro SOAP pomocí prostředí Eclipse” na stránce 928
   b) “Vyvíjení klienta JAX-WS pro přenos WebSphere pro protokol SOAP pomocí prostředí Eclipse” na stránce 949
   c) “Implementace klienta webové služby do prostředí Axis2 pro použití přenosu WebSphere MQ pro SOAP” na stránce 981
2. Připojte se k webové službě Axis StockQuotepomocí klienta Axis2 a doporučeného doporučení W3C pro protokol SOAP prostřednictvím rozhraní JMS.
   a) “Konfigurace prostředků produktu WebSphere MQ” na stránce 974
   b) “Konfigurace prostředků serveru WebSphere Application Server” na stránce 975
   c) “Vývoj webové služby JAX-WS EJB pro W3C SOAP prostřednictvím rozhraní JMS” na stránce 935
   d) “Implementace na klienta Axis2 pomocí protokolu W3C SOAP prostřednictvím rozhraní JMS” na stránce 984

Most WebSphere MQ pro protokol HTTP

Při použití mostu produktu WebSphere MQ pro protokol HTTP mohou aplikace klienta vyměňovat zprávy s produktem WebSphere MQ bez nutnosti instalovat klienta WebSphere MQ MQI. Produkt WebSphere MQ můžete volat z libovolné platformy nebo jazyka s možností HTTP.

Úvod k mostu produktu WebSphere MQ pro protokol HTTP

Most WebSphere MQ pro protokol HTTP je webová aplikace prostředí Java, Enterprise Environment (JEE). Klienti HTTP mohou odesílat, procházet a odstraňovat zprávy z front produktu WebSphere MQ , aby mohli odesílat, procházet a odstraňovat zprávy z produktu **POST**, **GET**a **DELETE** . Most produktu WebSphere MQ pro protokol HTTP není vhodný pro použití se zprávami, je-li vyžadováno zajištěné doručení.

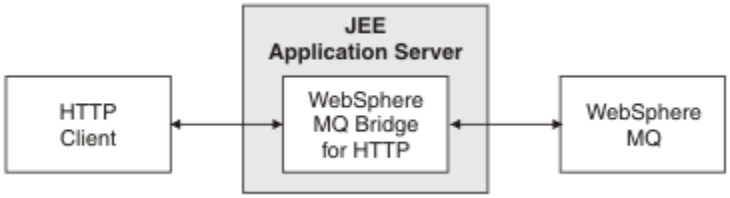
Výhody

Pomocí mostu WebSphere MQ pro protokol HTTP můžete odesílat a přijímat zprávy produktu WebSphere MQ pomocí protokolu HTTP ze širokého spektra prostředí:

- Prostředí, která podporují protokol HTTP, ale ne produkt WebSphere MQ.
- Prostředí, která nemají dostatek úložného prostoru k instalaci klienta WebSphere MQ MQI.
- Prostředí, které je příliš mnoho pro instalaci klienta WebSphere MQ MQI na každém systému, který vyžaduje přístup k produktu WebSphere MQ.
- Webové aplikace, ze kterých chcete odesílat nebo přijímat zprávy bez kódování vlastního mostu do produktu WebSphere MQ.
- Webové aplikace, které chcete vylepšit, pomocí asynchronních metod, jako je technologie AJAX. Produkt WebSphere MQ Bridge for HTTP zpřístupňuje fronty a témata produktu WebSphere MQ prostřednictvím služby Representation State Transfer (REST) prostřednictvím protokolu HTTP.

Podporu HTTP lze používat s dvoubodovým systémem zpráv i s topologií systému zpráv typu publikování-odběr.

Jak podpora HTTP funguje?



Obrázek 209. Most WebSphere MQ pro protokol HTTP

Most WebSphere MQ pro webovou aplikaci HTTP přijímá požadavky HTTP z jednoho nebo více klientů. Na jejich zastoupení interaguje s produktem WebSphere MQ a vrací odpovědi HTTP na ně.

Most produktu WebSphere MQ pro protokol HTTP je servlet JEE, který je připojen k produktu WebSphere MQ pomocí adaptéru prostředků. Servlet HTTP zpracovává tři různé typy požadavků HTTP: **POST**, **GET**a **DELETE**.

Tabulka 152. Most produktu WebSphere MQ pro příkazy HTTP

Požadavek HTTP	**Výsledek**
POST	Přepne zprávu do fronty nebo tématu.
GET	Prochází první zprávu ve frontě. V řádku s protokolem HTTP příkaz **GET** neodstraní zprávu z fronty. Nepoužívejte produkt **GET** s zasíláním zpráv publikování/odběru.
ODSTRANIT	Získá a odstraní zprávu z fronty nebo tématu.

Příklad HTTP POST

HTTP **POST** vloží zprávu do fronty nebo publikaci do tématu. Ukázka **HTTPPOST** Java je příkladem požadavku HTTP **POST** zprávy na frontu. Místo použití jazyka Java můžete vytvořit požadavek HTTP **POST** pomocí formuláře prohlížeče nebo sady nástrojů AJAX.

Obrázek 210 na stránce 991 ukazuje požadavek HTTP na vložení zprávy do fronty s názvem `myQueue`. Tento požadavek obsahuje záhlaví HTTP `x-msg-correlId` k nastavení ID korelace zprávy produktu WebSphere MQ .

```
POST /msg/queue/myQueue/ HTTP/1.1
Host: www.example.org
Content-Type: text/plain
x-msg-correlID: 1234567890
Content-Length: 50

Here is my message body that is posted on the queue.
```

*Obrázek 210. Příklad požadavku HTTP **POST** na frontu*

Obrázek 211 na stránce 991 zobrazuje odezvu odeslanou zpět klientovi. Není žádný obsah odezvy.

```
HTTP/1.1 200 OK
Date: Wed, 2 Jan 2007 22:38:34 GMT
Server: Apache-Coyote/1.1 WMQ-HTTP/1.1 JEE-Bridge/1.1
Content-Length: 0
```

Obrázek 211. Příklad odezvy HTTP POST

Příklad HTTP DELETE

HTTP **DELETE** získá zprávu z fronty a odstraní ji, nebo načte a odstraní publikaci. Ukázka **HTTPDELETE** Java je příkladem požadavku HTTP **DELETE**, který čte zprávu z fronty. Místo použití jazyka Java můžete vytvořit požadavek HTTP **DELETE** pomocí formuláře prohlížeče nebo sady nástrojů AJAX.

Produkt Obrázek 212 na stránce 992 je požadavkem HTTP na odstranění další zprávy ve frontě s názvem `myQueue`. V odezvě se na klienta vrátí tělo zprávy. Ve výrazech produktu WebSphere MQ je HTTP **DELETE** destruktivním získacím.

Požadavek obsahuje záhlaví požadavku HTTP `x-msg-wait`, které instruuje produkt WebSphere MQ Bridge pro protokol HTTP, jak dlouho čekat, než zpráva dorazí do fronty. Požadavek dále obsahuje záhlaví požadavku `x-msg-require-headers`, které určuje, že klient má v odezvě přijmout ID korelace zprávy.

```
DELETE /msg/queue/myQueue/ HTTP/1.1
Host: www.example.org
x-msg-wait: 10
x-msg-require-headers: correlID
```

*Obrázek 212. Příklad požadavku HTTP **DELETE***

Obrázek 213 na stránce 992je odezva vrácena klientovi. ID korelace se vrátí na klienta, jak požadovalo záhlaví požadavku `x-msg-require-headers`.

```
HTTP/1.1 200 OK
Date: Wed, 2 Jan 2007 22:38:34 GMT
Server: Apache-Coyote/1.1 WMQ-HTTP/1.1 JEE-Bridge/1.1
Content-Length: 50
Content-Type: text/plain; charset=utf-8
x-msg-correlId: 1234567890

Here is my message body that is retrieved from the queue.
```

*Obrázek 213. Příklad odezvy HTTP **DELETE***

Příklad HTTP GET

HTTP **GET** získá zprávu z fronty. Zpráva ve frontě zůstane. V termínech WebSphere MQ je HTTP **GET** požadavek na procházení. Požadavek HTTP **GET** můžete vytvořit pomocí klienta Java, formuláře prohlížeče nebo sady nástrojů AJAX.

Obrázek 214 na stránce 992 je požadavek HTTP na procházení další zprávy ve frontě s názvem `myQueue`.

Požadavek obsahuje záhlaví požadavku HTTP `x-msg-wait`, které instruuje produkt WebSphere MQ Bridge pro protokol HTTP, jak dlouho čekat, než zpráva dorazí do fronty. Požadavek dále obsahuje záhlaví požadavku `x-msg-require-headers`, které určuje, že klient má v odezvě přijmout ID korelace zprávy.

```
GET /msg/queue/myQueue/ HTTP/1.1
Host: www.example.org
x-msg-wait: 10
x-msg-require-headers: correlID
```

*Obrázek 214. Příklad požadavku HTTP **GET***

Obrázek 215 na stránce 993 je odezva vrácená klientovi. ID korelace se vrátí na klienta, jak požadovalo záhlaví požadavku `x-msg-require-headers`.

```
HTTP/1.1 200 OK
Date: Wed, 2 Jan 2007 22:38:34 GMT
Server: Apache-Coyote/1.1 WMQ-HTTP/1.1 JEE-Bridge/1.1
Content-Length: 50
Content-Type: text/plain; charset=utf-8
x-msg-correlId: 1234567890

Here is my message body that appears on the queue.
```

*Obrázek 215. Příklad odezvy HTTP **GET***

Instalace, konfigurace a ověření mostu produktu WebSphere MQ pro protokol HTTP

Získejte most produktu WebSphere MQ pro protokol HTTP pomocí instalace produktu "Systém zpráv Java Messaging a webové služby" z materiálů instalace klienta nebo serveru WebSphere MQ MQI. Implementujte produkt WebSphere MQ Bridge pro protokol HTTP na vhodný aplikační server.

Než začnete

Zkontrolujte předpokládané produkty na adrese Systémové požadavky pro produkt IBM WebSphere MQ. Proces instalace nekontroluje přítomnost a dostupnost nezbytného softwaru pro spuštění mostu produktu WebSphere MQ pro protokol HTTP. Musíte ověřit, zda jsou předpoklady nainstalovány.

Most produktu WebSphere MQ pro protokol HTTP lze spustit na libovolném aplikačním serveru Java EE 1.4 prostřednictvím instalace adaptéru prostředků WebSphere MQ . Můžete také spustit most produktu WebSphere MQ pro protokol HTTP na verzi produktu WebSphere Application Server starší než verze 6.0.2.1. Pomocí portu MLP (WebSphere Application Server Message Listener Port) integrujte produkt WebSphere MQ jako poskytovatele rozhraní JMS.

Podpora mostu produktu WebSphere MQ pro protokol HTTP je poskytována pouze pro následující aplikační servery:

- WebSphere Application Server 6.0.2.1 a novější.
- WebSphere Application Server Community Edition verze 1.1 a novější.

Informace o této úloze

Produkt WebSphere MQ Bridge for HTTP je dodáván jako soubor `.war` , `WMQHTTP.war`.

- Na platformách UNIX a Linux
  - `WMQHTTP.war` je součástí instalační volby "Java Messaging and Web Services" . Tato volba je k dispozici v obou instalačních materiálech klienta i serveru.
  - Produkt `WMQHTTP.war` je nainstalován do produktu *<mqmtop>*`/java/http/WMQHTTP.war`. *<mqmtop>* je adresář, do kterého je nainstalován produkt WebSphere MQ .
  - Produkt `WMQHTTP.samples` je nainstalován do produktu *<mqmtop>*`/java/http/samples`. *<mqmtop>* je adresář, do kterého je nainstalován produkt WebSphere MQ .

Proveďte následující kroky instalace a nainstalujte produkt WebSphere MQ Bridge for HTTP, implementujte jej a nakonfigurujte jej a ověřte konfiguraci. Podrobnosti o krocích konfigurace se liší na různých aplikačních serverech. Použijte “Implementace a ověření mostu produktu WebSphere MQ pro protokol HTTP na serveru WebSphere Application Server V6.1.0.9” na stránce 994 jako šablonu pro postup, jak postupovat na vašem aplikačním serveru.

Postup

1. Získejte produkt `WMQHTTP.war` pomocí instalace klienta nebo serveru WebSphere MQ MQI.
2. Zkopírujte produkt `WMQHTTP.war` na server, ze kterého jej lze implementovat na aplikační server.

3. Implementujte produkt `WMQHTTP.war` na aplikační server.
4. V případě potřeby nainstalujte produkt WebSphere MQ jako adaptér prostředků na aplikační server.

   Zjistěte, zda je produkt WebSphere MQ již konfigurován jako poskytovatel systému zpráv na aplikačním serveru. Použijte nástroj pro administraci nebo správu dodávaný s aplikačním serverem, abyste se podívali na produkt WebSphere MQ. WebSphere MQ může být nalezen pod touto cestou, **Prostředky > JMS > Poskytovatelé systému zpráv**.
5. Konfigurovat továrnu připojení na aplikačním serveru pro připojení ke správci front, který používá přenos klienta WebSphere MQ MQI[12].
6. Nakonfigurujte webovou aplikaci `WMQHTTP.war` na aplikačním serveru, aby používala továrnu připojení.
7. Ověřte konfiguraci.
   a) Nastavte správce front uvedeného v továrně připojení a v lokální frontě.
   b) Umístěte zprávu do lokální fronty.
   c) Vytvořte kanál připojení serveru uvedený v továrně připojení s oprávněním pro čtení a zápis do lokální fronty.
   d) Spusťte správce front a modul listener.
   e) Spusťte aplikační server a server `WMQHTTP.war`, pokud ještě nejsou spuštěni.
   f) Otevřete prohlížeč a zadejte příkaz `http://`*`hostname`*`:`*`web port`*`/`*`Context root`*`/msg/queue/`*`local queue`*`.`

Výsledky

V okně prohlížeče se zobrazí zpráva, kterou jste umístili do lokální fronty.

Jak pokračovat dále

1. Zkuste příklad, “Implementace a ověření mostu produktu WebSphere MQ pro protokol HTTP na serveru WebSphere Application Server V6.1.0.9” na stránce 994.
2. Spusťte ukázkové aplikace Java Java.

Implementace a ověření mostu produktu WebSphere MQ pro protokol HTTP na serveru WebSphere Application Server V6.1.0.9

Následující příklad slouží k přípravě implementace produktu WebSphere MQ Bridge pro protokol HTTP ke spuštění ukázkových programů HTTP jazyka Java. Implementace je na serveru WebSphere Application Server V6.1.0.9.

Než začnete

1. Postupujte podle pokynů v části “Instalace, konfigurace a ověření mostu produktu WebSphere MQ pro protokol HTTP” na stránce 993a zkopírujte produkt `WMQHTTP.war` na server, který je přístupný pro vaši instalaci serveru WebSphere Application Server.
2. Nakonfigurujte správce front a frontu, aby bylo možné použít testování konfigurace:
   - V tomto příkladu je správce front konfigurován s použitím hodnot v produktu Tabulka 153 na stránce 994:

Tabulka 153. Konfigurace správce front	
Objekt	**Hodnota**
Název hostitele	`itso-01`
Správce front	`QM1`

[12] Nejprve nakonfigurujte přenos klienta alespoň na začátku. Některé aplikační servery se mohou připojit k produktu WebSphere MQ prostřednictvím přímých připojení nebo připojení v režimu vazeb.

Tabulka 153. Konfigurace správce front (pokračování)	
Objekt	**Hodnota**
Lokální fronta	HTTPTESTQ
Kanál připojení serveru	MYSVRCON Nakonfigurujte jméno uživatele MCA s dostatečným oprávněním pro čtení a zápis do HTTPTESTQ.
Port modulu listener	1414

3. Spustit správce front a modul listener
4. Umístěte testovací zprávu do produktu HTTPTESTQ. Příklad:
   a. Spusťte produkt WebSphere MQ Explorer.
   b. V seznamu lokálních front pro QM1klepněte pravým tlačítkem myši na volbu **HTTPTESTQ > Vložit testovací zprávu > typ `First Message` > Vložit zprávu > Zavřít** .
5. Spusťte aplikační server a přihlaste se do konzoly Integrated Solutions Console.

Informace o této úloze

Tento příklad ukazuje kroky, které je třeba provést, pokud spouštíte server WebSphere Application Server V6.1.0.9 jako aplikační server. Pokud provozujete jinou verzi serveru WebSphere Application Server nebo spouštíte jiný aplikační server, jsou kroky odlišné. Produkt WebSphere Application Server V6.1.0.9 je předkonfigurován s produktem WebSphere MQ nainstalovaným jako poskytovatel zpráv pomocí knihoven klienta WebSphere MQ MQI. Pokud produkt WebSphere MQ není předkonfigurován jako poskytovatel systému zpráv nebo pokud chcete použít vazby serveru WebSphere MQ , je třeba instalovat a konfigurovat adaptér prostředků WebSphere MQ pro prostředí JEE na aplikační server.

Postupujte podle pokynů pro implementaci mostu produktu WebSphere MQ pro protokol HTTP na server WebSphere Application Server V6.1.0.9a ověřte implementaci pomocí prohlížeče:

Postup

1. V navigačním podokně klepněte na volbu **Prostředky > Poskytovatelé JMS > Poskytovatel systému zpráv WebSphere MQ**.

   Můžete nakonfigurovat buď na úrovni uzlu, buňky nebo serveru, v závislosti na implementaci serveru WebSphere Application Server. V příkladu je použita implementace na úrovni serveru.
2. V části **Další vlastnosti**klepněte na volbu **Továrny na připojení > Nový**.
3. Ve formuláři Poskytovatelé JMS zadejte informace v produktu Tabulka 154 na stránce 995nebo alternativy dle vašeho výběru, klepněte na tlačítko **Použít > Uložit**.

Tabulka 154. Nastavit nebo upravit následující pole	
Pole	**Hodnota**
Název	WMQHTTPBridge
Název rozhraní JNDI	jms/WMQHTTPJCAConnectionFactory
Správce front	QM1
Hostitel	itso-01
Port	1414
Kanál	MYSVRCON
Typ přenosu	CLIENT

4. V navigačním podokně klepněte na volbu **Aplikace > Instalovat novou aplikaci**.

5. Vložte cestu k produktu `WMQHTTP.war` do formuláře a zadejte kontextový kořenový adresář a klepněte na tlačítko **Další**.
   a) Kontextový kořenový adresář je volitelný. `mq` je výchozí kontextový kořenový adresář pro ukázkové aplikace HTTP.
   b) Kontextový kořenový adresář je součástí identifikátoru URI identifikujícího most produktu WebSphere MQ pro protokol HTTP. Můžete vynechat kontextový kořenový adresář, nebo jej později změnit.
6. Na stránce **Vybrat volby instalace** průvodce instalací nemusíte měnit žádné výchozí nastavení, klepněte na tlačítko **Další**.
7. Na stránce **Mapovat moduly na servery** vyberte klastr nebo Server, zaškrtněte políčko Select a klepněte na tlačítko **Použít > Další**.
8. Na stránce **Mapovat odkazy na prostředky na prostředky** ve formuláři **javax.jms.ConnectionFactory** klepněte na tlačítko **Procházet ...** na mostu IBM WebSphere MQ pro řádek HTTP.
9. Na stránce **Podnikové aplikace > Dostupné prostředky** vyberte volbu **WMQHTTPBridge**a klepněte na tlačítko **Použít**.
10. Zpět ve formuláři **javax.jms.ConnectionFactory** vyberte metodu ověření.
    a) V případě příkladu vyberte volbu **Žádná**a klepněte na tlačítko **Použít**. Další volby vyžadují další konfiguraci.
11. Zaškrtněte políčko **Vybrat** u mostu IBM WebSphere MQ pro protokol HTTP a poté klepněte na tlačítko **Další > Další > Dokončit > Uložit** .
12. V navigačním podokně klepněte na volbu **Aplikace > Podnikové aplikace**.
13. Zaškrtněte políčko pro výběr produktu `WMQHTTP.wara` klepněte na volbu **Spustit**.
14. Otevřete okno prohlížeče. Zadejte `http://itso-01:9080/mq/msg/queue/HTTPTESTQ`s použitím odpovídajícího názvu hostitele a portu.

Výsledky

Pokud je konfigurace úspěšná, zobrazí se okno prohlížeče `First Message`.

Jak pokračovat dále

Spusťte ukázkové aplikace Java Java.

Publikování/odběr pomocí mostu produktu WebSphere MQ pro protokol HTTP

Produkt WebSphere MQ Bridge for HTTP používá třídy WebSphere MQ pro rozhraní JMS publish/subscribe. HTTP **POST** vytvoří publikaci. Protokol HTTP **DELETE** vytvoří netrvalý spravovaný odběr. Před použitím identifikátoru URI tématu je třeba nakonfigurovat publikování/odběr pro službu JMS.

Publikování/odběr je plně integrován do produktu WebSphere MQ ve verzi 7. Před verzí 7 se v rámci samostatného zprostředkovatele publikování/odběru zpracují publikace a odběry. Nazývá se "řazen do fronty" publikovat/odebírat, aby se odlišil od plně integrovaného publikování/odběru ve verzi 7. Verze 7 emuluje ve frontě publikování ve frontě s využitím integrovaného publikování/odběru. Emulace umožňuje, aby existující aplikace zařazené do fronty publikování/odběru existovaly společně s integrovanými aplikacemi spuštěnými ve stejném správci front. Aplikace typu publikování/odběr zařazené do fronty mohou také spolupracovat s integrovanými aplikacemi a sdílet stejná témata. Ve verzi 6 byl zprostředkovatel dodáván s produktem WebSphere MQ; před verzí 6 byl k dispozici jako sada SupportPack.

Konfigurace

Produkt WebSphere MQ Bridge for HTTP používá rozhraní JMS k publikování a odběru. Ve verzi 7 můžete řídit, zda třídy WebSphere MQ pro službu JMS používají ve frontě nebo integrované publikování/odběr, pomocí vlastnosti JMS produktu `PROVIDERVERSION` .

Další úvaha je, že můžete použít buď knihovny klienta WebSphere MQ MQI s rozhraním WebSphere MQ Bridge for HTTP, nebo knihovny serveru. Klientské knihovny verze 6 podporují pouze ve frontě publikování/odběr, zatímco knihovny verze 7 podporují zařazení do fronty i integrované publikování/odběr. Většina webových nebo aplikačních serverů, které používají produkt WebSphere MQ jako poskytovatele systému zpráv, používají knihovny klienta. Chcete-li používat integrované publikování/odběr, musí být jak klient WebSphere MQ MQI, tak i knihovny serveru alespoň ve verzi 7. Pokud je spuštěna dřívější verze produktu WebSphere než 7, musíte nakonfigurovat publikování/subscribe; ve frontě. Další informace najdete v tématu Tabulka 155 na stránce 997. Zkontrolujte, které knihovny jsou nainstalovány nebo nakonfigurovány s webovým serverem nebo aplikačním serverem, který používáte.

Tabulka 155. Režimy konfigurace publikování a odběru

	Klient V6 nebo starší	**Klient V7 nebo pozdější**
Server V6 nebo starší	1. Spusťte skript `\java\bin\MQJMS_PSQ.mqsc` .	Nepodporováno
Server V7 nebo pozdější	1. Spusťte skript `\java\bin\MQJMS_PSQ.mqsc` . 2. Nastavit správce front na `PSMODE=ENABLED`	1. Pokud PROVIDERVERSION = 7 a. Nastavit správce front na `PSMODE=ENABLED` nebo `PSMODE=COMPAT` 2. Pokud PROVIDERVERSION = 6 a. Nastavit správce front na `PSMODE=ENABLED`

Publikovat

Odešlete požadavek HTTP **POST** s identifikátorem URI:

```
http://hostname:port/context_root/msg/topic/topicString
```

Obsahy zpráv se publikují za použití řetězce tématu *`topicString`*.

Odebírat

Odešlete požadavek HTTP **DELETE** s identifikátorem URI:

```
http://hostname:port/context_root/msg/topic/topicString
```

Most produktu WebSphere MQ pro protokol HTTP vytváří spravovaný netrvalý odběr pro řetězec tématu *`topicString`*. Odběr je odstraněn, jakmile je vrácena publikace, nebo do vypršení intervalu čekání nastaveného vlastním záhlavím entity `x-msg-wait`.

Spuštění mostu produktu WebSphere MQ pro ukázky HTTP

Most produktu WebSphere MQ pro ukázky HTTP je k dispozici pouze pro použití v operačním systému Windows . Ukázky ukazují, jak odeslat příkazy HTTP **POST** a HTTP **DELETE** do mostu WebSphere MQ pro protokol HTTP z programů v jazyce Java.

Než začnete

Ověřte svůj most produktu WebSphere MQ pro instalaci pomocí protokolu HTTP spuštěním kroku “7” na stránce 994 v tématu “Instalace, konfigurace a ověření mostu produktu WebSphere MQ pro protokol HTTP” na stránce 993.

Ukázky HTTP se nainstalují do adresářů zobrazených v Tabulka 156 na stránce 998. V každém případě je zdrojový kód nainstalován do podadresáře `/src` .

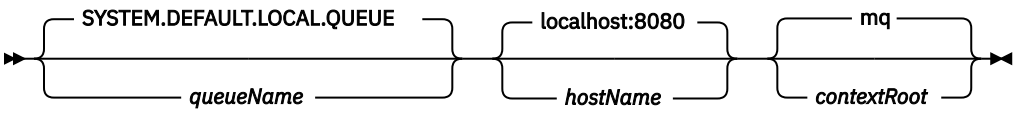
Tabulka 156. Umístění ukázek HTTP	
Platforma	**Umístění**
Windows	*MQ_INSTALLATION_PATH*/tools/http/samples
Všechny ostatní platformy	*MQ_INSTALLATION_PATH*/samp/http
MQ_INSTALLATION_PATH představuje adresář, kde je nainstalován produkt WebSphere MQ .	

Informace o této úloze

Ukázky simulují ukázkové aplikace WebSphere MQ AMQSPUT a AMQSGET. Vykreslují následující funkce v prostředí systému zpráv typu point-to-point:

- **HTTPPOST** -Odesílá požadavky HTTP **POST** v aplikaci v jazyce Java pro vložení zpráv do fronty WebSphere MQ pomocí mostu WebSphere MQ pro protokol HTTP a zpracovává odpovědi.
- **HTTPDELETE** -Odesílá požadavky HTTP **DELETE** v aplikaci Java k získání zpráv z fronty WebSphere MQ pomocí mostu WebSphere MQ pro protokol HTTP a zpracovává odpovědi obsahující zprávu produktu WebSphere MQ .

Parametry pro HTTPPOST a HTTPDELETE



Chcete-li spustit ukázku **HTTPPOST** , proveďte následující kroky:

Postup

1. V příkazovém okně přejděte do adresáře ukázek HTTP.
2. Spusťte ukázku **HTTPPOST** .

   ```
   java -classpath . HTTPPOST [parameters]
   ```

   Když se spustí ukázka **HTTPPOST** , zobrazí se následující výstup:

   ```
   HTTP POST Sample start
   Target server is 'hostName'
   Target queue is 'queueName'
   Target context-root is 'contextRoot'
   ```

3. Do příkazového řádku zadejte text, který má tvořit tělo zprávy.
4. Chcete-li tuto zprávu odeslat do fronty produktu WebSphere MQ , stiskněte klávesu Enter.
   a) Chcete-li odeslat další zprávu, zadejte nějaký další text.
      Texty formuláře tvoří tělo druhé zprávy produktu WebSphere MQ .
   b) Chcete-li tuto zprávu odeslat do fronty produktu WebSphere MQ , stiskněte klávesu Enter.
5. Stiskněte dvakrát klávesu Enter, abyste ukončili **HTTPPOST**.
   Zobrazí se následující výstup:

   ```
   HTTP POST Sample end
   ```

Jak pokračovat dále

Ukázka **HTTPDELETE** provádí destruktivní získání všech zpráv, které jste umístili na frontu WebSphere MQ .

Spusťte ukázku produktu **HTTPDELETE** provedením následujících kroků:

1. V okně příkazového řádku přejděte na *MQ_INSTALLATION_PATH*/tools/samples. *MQ_INSTALLATION_PATH* Představuje adresář, kde je nainstalován produkt WebSphere MQ .
2. Spusťte ukázku **HTTPDELETE** .

   ```
   java -classpath . HTTPPOST [parameters]
   ```

   Když se spustí ukázka **HTTPDELETE** , zobrazí se následující výstup:

```
HTTP DELETE Sample start
Target server is 'host:port'
Target queue is 'your queue name'
Target context-root is 'your context-root'
message
message
...
```

Aspekty zabezpečení mostu WebSphere pro protokol HTTP

Pro ověření klienta webového prohlížeče platí standardní aspekty zabezpečení webu. Oprávnění k prostředkům produktu WebSphere MQ je na úrovni uživatele, který spouští most WebSphere Bridge pro HTTP servlet, a nikoli pro jednotlivé klienty webového prohlížeče. Standardní zabezpečení produktu WebSphere MQ se vztahuje na produkt WebSphere MQ.

Data proudící z webového prohlížeče do aplikace WebSphere MQ pomocí mostu WebSphere pro protokol HTTP a back-to činí tři kroky:

Připojení klienta
Od prohlížeče k produktu WebSphere Bridge for HTTP přes připojení TCP/IP pomocí protokolu HTTP.

Připojení adaptéru prostředků k produktu WebSphere MQ
Připojení je z mostu produktu WebSphere pro protokol HTTP se správcem front produktu WebSphere MQ . Připojení je buď připojení klienta, přes TCP/IP, nebo lokální připojení vazeb WebSphere MQ . Po navázání připojení je požadavek HTTP umístěn na standardní lokální frontu nebo na přenosovou frontu.

Z lokální fronty produktu WebSphere MQ přes jeden nebo více kanálů do cílové fronty.
Použití standardních technik pro zabezpečení front, témat, správců front a kanálů.

Odpověď podnivá kroky opačně.

Připojení klienta

Zabezpečená připojení mezi klienty HTTP a aplikačním serverem pomocí webového kontejneru. Použít standardní metody serveru HTTP, jako např. použití HTTPS. Informace naleznete v dokumentaci k aplikačnímu serveru.

Připojení adaptéru prostředků k produktu WebSphere MQ

Připojení mezi adaptérem prostředků a správcem front je autorizováno pouze s použitím jednoho ID uživatele. Přiřaďte ID jednoho uživatele k identifikaci požadavků z produktu WebSphere Bridge for HTTP. ID uživatele musí mít omezené autorizace produktu WebSphere MQ pouze pro externí uživatele prostředků, k jejichž přístupu musí být přístup. Musíte ověřit skutečného klienta samostatně a navázat důvěru pro následné interakce s klientem pomocí standardních technik pro webové zabezpečení.

Zabezpečte připojení mezi adaptérem prostředků a správcem front pomocí jediného ID uživatele. Omezte oprávnění, které ID uživatele nemá, aby více než bylo potřeba ke čtení a zápisu zpráv do front a témat. Produkt WebSphere Bridge for HTTP je bod pro útok mezi internetem a intranet.

Způsob zabezpečení připojení mezi adaptérem prostředků a produktem WebSphere MQ je závislý na konkrétním adaptéru prostředků. Pročtěte si dokumentaci pro adaptér prostředků.

Použití rozhraní modelu objektu Component Interface (WebSphere MQ Automation Classes for ActiveX)

Produkt WebSphere MQ Automation Classes for ActiveX (MQAX) jsou komponenty ActiveX , které poskytují třídy, které můžete použít ve své aplikaci k přístupu k produktu WebSphere MQ.

Produkt MQAX vyžaduje prostředí produktu WebSphere MQ a odpovídající aplikaci produktu WebSphere MQ , se kterou se má komunikovat.

Poskytuje vám aplikaci ActiveX schopnost spouštět transakce a přistupovat k datům na libovolném z vašich podnikových systémů, k jejichž přístupu můžete přistupovat prostřednictvím produktu WebSphere MQ.

WebSphere MQ Automation Classes for ActiveX:

- Poskytnutí přístupu k funkcím a funkcím rozhraní API produktu WebSphere MQ umožňuje úplnou vzájemnou konektivitu s jinými platformami WebSphere MQ .
- Přepočítá se s normálními konvencemi, které se očekávají od komponenty ActiveX .
- Konform k modelu objektu WebSphere MQ , který je také k dispozici pro prostředí .NET, C + +, Java a LotusScript.

Jsou poskytnuty ukázky spouštěče MQAX. Tyto ukázky můžete použít nejprve ke kontrole, zda je instalace produktu MQAX úspěšná a zda máte nainstalované základní prostředí produktu WebSphere MQ . Ukázky také ukazují, jak lze použít MQAX.

Skriptování COM a ActiveX

Model COM (Component Object Model) je objektový model založený na objektu Microsoftdefinovaný modelem. Uvádí, jak mohou být softwarové komponenty poskytovány způsobem, který jim umožňuje lokalizovat a komunikovat mezi sebou bez ohledu na jazyk počítače, ve kterém jsou napsány, nebo na jejich umístění.

ActiveX je sada technologií založených na COM, které integruje vývoj aplikací, znovupoužitelné komponenty a internetové technologie na platformách Microsoft Windows . Komponenty ActiveX poskytují rozhraní, ke kterým lze dynamicky přistupovat pomocí aplikací. Skriptovací klient ActiveX je aplikace, například kompilátor, který může sestavit nebo spustit program nebo skript, který používá rozhraní poskytovaná komponentami ActiveX (nebo COM).

Podpora prostředí produktu WebSphere MQ

WebSphere MQ Automation Classes for ActiveX lze použít pouze s **32bitovými** skriptovacími klienty ActiveX .

Komponentu COM lze použít pouze pro aplikace **32-bit** . Chcete-li zapsat 64bitovou aplikaci COM, můžete použít rozhraní .NET.

Chcete-li spustit produkt MQAX v prostředí serveru WebSphere MQ , musíte mít v systému nainstalován operační systém Windows 2000 nebo novější.

Chcete-li spustit produkt MQAX v prostředí klienta WebSphere MQ MQI, je třeba v systému Windows 2000 nebo novějším ve vašem systému instalovat klienta WebSphere MQ MQI:

Klient WebSphere MQ MQI vyžaduje přístup k alespoň jednomu serveru WebSphere MQ . Je-li na vašem systému nainstalován klient WebSphere MQ MQI i server WebSphere MQ , budou aplikace MQAX vždy spuštěny vůči serveru. Rozhraní ActiveX k rozhraní MQAI je k dispozici pouze v prostředí serveru WebSphere MQ .

Návrh a programování s použitím tříd automatizace produktu WebSphere MQ pro ActiveX

Návrh aplikací MQAX, které přistupují k jiným aplikacím nežActiveX

Třídy automatizace produktu WebSphere MQ poskytují přístup k funkcím rozhraní API produktu WebSphere MQ . Výhody plynoucí z použití produktu WebSphere MQ do vaší aplikace systému Windows proto můžete využít ve všech výhodách.

Celkový návrh vaší aplikace je stejný jako pro všechny aplikace produktu WebSphere MQ , takže zvažte všechny aspekty návrhu popsané v sekci “Vývoj aplikací” na stránce 7 .

Chcete-li používat třídy automatizace produktu WebSphere MQ , kódujete programy systému Windows ve vaší aplikaci s použitím jazyka, který podporuje vytváření a používání objektů COM. Příklad: Visual Basic, Java a další skriptovací klienti ActiveX . Třídy lze poté snadno integrovat do vaší aplikace, protože objekty produktu WebSphere MQ , které potřebujete, lze kódovat pomocí nativní syntaxe jazyka implementace.

Použití tříd automatizace produktu WebSphere MQ pro ActiveX

Při návrhu aplikace ActiveX , která používá třídy WebSphere MQ Automation Classes for ActiveX, je nejdůležitější položkou informací zpráva, která je odeslána nebo přijata ze vzdáleného systému WebSphere MQ . Proto musíte znát formát položek, které jsou vloženy do zprávy. Pro skript MQAX k určité práci musí znát strukturu zprávy i aplikaci produktu WebSphere MQ , která tuto zprávu vyzvedne nebo odešle.

Pokud odesíláte zprávu s aplikací MQAX a chcete provést převod dat na konci struktury MQAX, musíte také vědět:

- Kódová stránka použitá vzdáleným systémem
- Kódování použité vzdáleným systémem

Chcete-li uchovat svůj kód přenosný, je dobrým zvykem nastavit kódovou stránku a kódování, i když jsou v současné době stejné jak v odesílajícím, tak v přijímajícím systému.

Při zvažování, jak strukturovat implementaci systému, který navrhujete, pamatujte na to, že se skripty MQAX spouštějí na stejném počítači jako ten, na kterém je nainstalován správce front WebSphere MQ nebo klient WebSphere MQ .

Rady a tipy pro programování

Následující rady a tipy nejsou ve významném pořadí. Jsou to témata, která vám mohou ušetřit čas, pokud jsou důležitá pro práci, kterou děláte.

Vlastnosti deskriptoru zpráv

Pokud manipulujete s vlastnostmi deskriptoru zpráv v programu, může být lepší použít hexadecimální ekvivalenty polí.

Informace v této části odkazují na následující vlastnosti:

- AccountingToken
- CorrelationId
- GroupId
- MessageId

Je-li aplikací produktu WebSphere MQ původcem zprávy a produkt WebSphere MQ tyto vlastnosti generuje, je lepší použít vlastnosti AccountingTokenHex, CorrelationIdHex, GroupIdHex a MessageIdHex, pokud se chcete podívat na jejich hodnoty nebo s nimi pracovat jakýmkoli způsobem, včetně jejich předání ve zprávě do produktu WebSphere MQ. Důvodem pro tuto skutečnost je, že generované hodnoty

produktu WebSphere MQ jsou řetězce v bajtech, které mají libovolnou hodnotu od 0 do 255 včetně, nejsou to řetězce tisknutelných znaků.

Kde je váš skript MQAX původcem zprávy, můžete použít buď vlastnosti AccountingToken, CorrelationId, GroupIda MessageId nebo jejich hexadecimální ekvivalenty.

Konstanty produktu WebSphere MQ

Konstanty produktu WebSphere MQ jsou k dispozici jako členové výčtu WebSphere MQ v knihovně MQAX200.

Řetězcové konstanty produktu WebSphere MQ

Řetězcové konstanty produktu WebSphere MQ a jejich odpovídající znakové řetězce.

Řetězcové konstanty produktu WebSphere MQ nejsou k dispozici při použití tříd automatizace produktu WebSphere MQ pro ActiveX. Musíte použít explicitní znakový řetězec pro ty, které jsou uvedeny v následujícím seznamu a všechny ostatní, které byste mohli potřebovat. Tyto příkazy musí být doplněny na osm znaků pomocí mezer:

MQFMT_NONE	" "
MQFMT_ADMIN	"MQADMIN"
MQFMT_CHANNEL_COMPLETED	"MQCHCOM"
MQFMT_CICS	"MQCICS"
MQFMT_COMMAND_1	"MQCMD1 "
MQFMT_COMMAND_2	"MQCMD2 "
HLAVIČKA MQFMT_DEAD_LETTER_HEADER	"MQDEAD"
ZÁHLAVÍ MQFMT_DICT_HEADER	"MQHDIST"
UDÁLOST MQFMT_EVENT	"MQEVENT"
MQFMT_IMS	"MQIMS"
Funkce MQFMT_IMS_VAR_STRIsNG	"MQIMSVS"
ROZŠÍŘENÍ MQFMT_MD_EXTENSION	"MQHMDE"
MQFMT_PCF	"MQPCF"
MQFMT_REF_MSG_HEADER	"MQHREF"
ZÁHLAVÍ MQFMT_RF_HEADER	"MQHRF"
ŘETĚZEC MQFMT_STRING	"MQSTR"
SPOUŠTĚČ MQFMT_TRIGGER	"MQTRIG"
MQFMT_WORK_INFO_HEADER	"MQHWIH"
ZÁHLAVÍ MQFMT_XMIT_Q_HEADER	"MQXMIT"

Konstanty řetězce null

Konstanty WebSphere MQ použité pro inicializaci čtyř vlastností MQMessage, MQMI_NONE (24 NULL znaků), MQCI_NONE (24 NULL znaků), MQGI_NONE (24 NULL znaků) a MQACT_NONE (32 NULL znaků), nejsou podporovány třídami automatizace WebSphere MQ pro ActiveX. Nastavení na prázdné řetězce má stejný efekt.

Chcete-li například nastavit různá ID zprávy MQMessage na tyto hodnoty: *mymessage*.**MessageId** = "" *mymessage*.**CorrelationId** = "" *mymessage*.**AccountingToken** = ""

Příjem zprávy z produktu WebSphere MQ

Existuje několik způsobů přijetí zprávy z produktu WebSphere MQ:

- Systém výzev pomocí funkce GET následovaný funkcí TIMER s použitím funkce Vizuální Basic TIMER.
- Zadání volby GET s volbou Wait; zadáte dobu čekání nastavením vlastnosti WaitInterval . Zvažte tuto možnost, i když jste nastavili systém, aby se spouštěl v prostředí s více vlákny, software, který běží v daném okamžiku, může běžet pouze s jedním vláknem. To zabrání tomu, aby se systém zamyká neomezeně.

  Ostatní vlákna nejsou ovlivněna. Pokud však ostatní podprocesy vyžadují přístup k produktu WebSphere MQ, vyžadují druhé připojení k produktu WebSphere MQ pomocí dalších objektů správce front MQAX a objektů front.

Zadání příkazu GET s volbou Wait a nastavením parametru WaitInterval na MQWI_UNLIMITED způsobí, že se systém zamkne až do dokončení volání GET, pokud je proces jediným vláknem.

Použití konverze dat

Dvě formy převodu dat jsou podporovány třídami automatizace produktu WebSphere MQ pro ActiveX -numerického kódování a konverze znakové sady.

Numerické kódování

Nastavíte-li vlastnost MQMessage Encoding, následující metody se převedou mezi různými systémy kódování čísel:

- Metoda ReadDecimal2
- Metoda ReadDecimal4
- Metoda ReadDouble
- Metoda ReadDouble4
- Metoda ReadFloat
- Metoda ReadInt2
- Metoda ReadInt4
- Metoda ReadLong
- Metoda ReadShort
- Metoda ReadUInt2
- Metoda WriteDecimal2
- Metoda WriteDecimal4
- Metoda WriteDouble
- Metoda WriteDouble4
- Metoda WriteFloat
- Metoda WriteInt2
- Metoda WriteInt4
- Metoda WriteLong
- Metoda WriteShort
- Metoda WriteUInt2

Vlastnost Encoding může být nastavena a interpretována pomocí dodaných konstant WebSphere MQ . Obrázek 216 na stránce 1004 uvádí příklad těchto:

```
/* Encodings for Binary Integers */
        MQENC_INTEGER_UNDEFINED
        MQENC_INTEGER_NORMAL
        MQENC_INTEGER_REVERSED

/* Encodings for Decimals */
        MQENC_DECIMAL_UNDEFINED
        MQENC_DECIMAL_NORMAL
        MQENC_DECIMAL_REVERSED

/* Encodings for Floating-Point Numbers */
        MQENC_FLOAT_UNDEFINED
        MQENC_FLOAT_IEEE_NORMAL
        MQENC_FLOAT_IEEE_REVERSED
        MQENC_FLOAT_S390
```

Obrázek 216. Zadané konstanty produktu WebSphere MQ pro kódování

Chcete-li například odeslat celé číslo ze systému Intel do operačního systému System/390 v kódování System/390 , postupujte takto:

```
Dim msg As New MQMessage   'Define a WebSphere MQ message for our use..
Print msg.Encoding    'Currently 546 (or X'222')
                                          'Set the encoding property
                                          to 785 (or X'311')
msg.Encoding = MQENC_INTEGER_NORMAL OR MQENC_DECIMAL_NORMAL
                         OR MQENC_FLOAT_S390
Print msg.Encoding    'Print it to see the change
Dim local_num As long 'Define a long integer
local_num = 1234                          'Set it
msg.WriteLong(local_num)         'Write the number into the message
```

Převod znakové sady

Konverze znakové sady je nezbytná, když posíláte zprávu z jednoho systému do jiného systému, kde jsou kódové stránky odlišné. Převod kódové stránky se používá:

- Metoda ReadString
- Metoda ReadNullTerminatedString
- Metoda WriteString
- Metoda WriteNullTerminatedString
- Vlastnost MessageData

Vlastnost MQMessage CharacterSet musí být nastavena na podporovanou hodnotu znakové sady (CCSID).

WebSphere MQ Automation Classes for ActiveX používá převodní tabulky k provedení konverze znakové sady.

Chcete-li například převést řetězce automaticky na kódovou stránku 437:

```
Dim msg As New MQMessage                'Define a WebSphere MQ message
msg.CharacterSet = 437                'Set code page required
msg.WriteString "A character string"'Put character string in message
```

Metoda WriteString přijímá řetězcová data ("Řetězec znaků" v příkladu) jako řetězec Unicode. Pak převede tato data z Unicode do kódové stránky 437 pomocí převodní tabulky 34B001B5.TBL.

Znaky v řetězci Unicode, které nejsou podporovány kódovou stránkou 437, jsou na kódové stránce 437 poskytnuty standardnímu substitučním znaku.

Podobně při použití metody ReadString má příchozí zpráva znakovou sadu vytvořenou hodnotou MQMD (WebSphere MQ Message Descriptor) a před předáním zpět do skriptovacího jazyka převod z této kódové stránky do znakové sady Unicode.

Využití vláken

WebSphere MQ Automation Classes for ActiveX implementuje model s podporou podprocesů, ve kterém mohou být objekty používány mezi podprocesy.

Přestože produkt MQAX povoluje použití objektů MQQueue a MQQueueManager , produkt WebSphere MQ v současné době nepovoluje sdílení manipulátorů mezi různými podprocesy.

Pokusy o použití těchto objektů v jiném podprocesu způsobí chybu a produkt WebSphere MQ vrátí návratový kód MQRC_HCONN_ERROR.

Poznámka: Pro každý proces existuje pouze jeden objekt MQSession. Použití relace MQSession CompletionCode a ReasonCode se nedoporučuje ve vícevláknových prostředích. Hodnoty chyb MQSession mohou být přepsány druhým vláknem mezi chybou, která byla vyvolána a kontrolována na prvním vláknu. Podprocesy jsou serializovány po dobu trvání každého volání metody nebo přístupu k vlastnosti. Takže zadáním příkazu Get s volbou Wait způsobí, že další podprocesy, které přistupují k objektům MQAX, budou pozastaveny, dokud nebude operace dokončena.

Ošetření chyb

Tyto informace popisují vlastnosti objektu MQAX, jak ošetřování chyb funguje, pravidla popisující způsob zpracování výjimek a získání vlastnosti.

Každý objekt MQAX obsahuje vlastnosti obsahující informace o chybě a metodu resetování nebo vymazání těchto vlastností. Vlastnosti jsou:

- CompletionCode
- ReasonCode
- ReasonName

Metoda je:

- Kódy ClearError

Jak funguje ošetřování chyb

Váš skript nebo aplikace MQAX vyvolá metodu objektu MQAX nebo přístupy nebo aktualizace vlastnosti objektu MQAX:

1. Aktualizují se kódy ReasonCode a CompletionCode v příslušném objektu.
2. Aktualizují se také ReasonCode a CompletionCode v objektu MQSession se stejnými informacemi.

   Poznámka: Omezení týkající se použití chybových kódů MQSession v aplikacích s podporou podprocesů viz “Využití vláken” na stránce 1005 .

   Je-li kód CompletionCode větší nebo roven vlastnosti ExceptionThreshold objektu MQSession, funkce MQAX vyvolá výjimku (číslo 32000). Použijte jej v rámci skriptu pomocí příkazu On-Chyba (nebo ekvivalentního příkazu), který má být zpracován.
3. Pomocí funkce Error můžete načíst přidružený chybový řetězec, který má tvar:

   ```
   MQAX: CompletionCode=xxx, ReasonCode=xxx, ReasonName=xxx
   ```

   Další informace o tom, jak používat chybové příkazy, naleznete v dokumentaci k skriptovacímu jazyku ActiveX .

   Použití položky CompletionCode a ReasonCode v objektu MQSession je vhodné pro jednoduché obslužné rutiny chyb.

   Vlastnost ReasonName vrací symbolický název produktu WebSphere MQ pro aktuální hodnotu prvku ReasonCode.

Vyvolání výjimek

Následující pravidla popisují, jak se zachází s výjimkami:

- Kdykoli vlastnost nebo metoda nastaví kód dokončení na hodnotu větší než nebo rovnou prahové hodnotě výjimky (obvykle je nastavena na 2), dojde k výjimce.
- Všechny volání metod a sady vlastností nastavují kód dokončení.

Získání vlastnosti

Jedná se o speciální případ, protože CompletionCode a ReasonCode nejsou vždy aktualizovány:

- Je-li vlastnost úspěšná, objekt a objekt MQSession ReasonCode a CompletionCode zůstávají nezměněny.
- Pokud vlastnost get selže s hodnotou CompletionCode varování, zůstane ReasonCode a CompletionCode nezměněno.
- Pokud vlastnost get selže s chybou CompletionCode , aktualizují se hodnoty ReasonCode a CompletionCode tak, aby odrážely skutečné hodnoty, a zpracování chyb bude pokračovat podle popisu.

Třída MQSession má metodu *ReasonCodeName* , která může být použita k nahrazení kódu příčiny WebSphere MQ symbolickým názvem. To je obzvláště užitečné při vývoji programů, v nichž může dojít k neočekávaným chybám. Název však není ideální pro prezentaci uživatelům.

Každá třída má také vlastnost *ReasonName*, která vrací symbolický název aktuálního kódu příčiny pro danou třídu.

WebSphere MQ Automation Classes for ActiveX -odkaz

Tento oddíl popisuje třídy produktu WebSphere MQ Automation Classes for ActiveX (MQAX), vyvinuté pro ActiveX. Třídy umožňují psát aplikace ActiveX , které mohou přistupovat k dalším aplikacím spuštěným v jiných prostředích nežActiveX , pomocí produktu WebSphere MQ.

Rozhraní WebSphere MQ Automation Classes for ActiveX

WebSphere MQ Automation Classes for ActiveX poskytuje předdefinované číselné konstanty ActiveX (jako např. MQMT_REQUEST), které jsou potřebné pro použití tříd.

Automatizační třídy ActiveX se skládají z následujících položek:

- “Třída MQSession” na stránce 1008
- “Třída MQQueueManager” na stránce 1011
- “Třída MQQueue” na stránce 1022
- “Třída MQMessage” na stránce 1036
- “Třída voleb MQPutMessage” na stránce 1057
- “Třída voleb MQGetMessage” na stránce 1059
- “Třída MQDistributionList” na stránce 1062
- “Třída položek MQDistributionList” na stránce 1066

Kromě toho produkt WebSphere MQ Automation Classes for ActiveX poskytuje předdefinované číselné konstanty ActiveX (jako např. MQMT_REQUEST), které jsou potřebné pro použití tříd. Ty jsou poskytovány ve výčtu MQ v knihovně MQAX200. Konstanty jsou podmnožinou těch, které jsou definovány v souborech záhlaví WebSphere MQ C (cmqc * .h) s některými dalšími třídami automatizace WebSphere MQ pro kódy příčiny ActiveX .

Informace o třídách automatizace produktu WebSphere MQ pro třídy ActiveX

Přečtěte si tyto informace spolu s referenčními tématy v části Vývoj odkazů na aplikace.

Důležité informace naleznete v příručce Funkce, které lze použít pouze s primární instalací na systému Windows .

Třída MQSession poskytuje kořenový objekt, který obsahuje stav poslední akce provedené na kterémkoli z objektů MQAX. Další informace viz “Ošetření chyb” na stránce 1005.

Třídy MQQueueManager a MQQueue poskytují přístup k základním objektům produktu WebSphere MQ . Metody nebo přístupy vlastností pro tyto třídy obecně vedou k volání napříč produktem WebSphere MQ MQI.

Třídy voleb MQMessage, MQPutMessagea MQGetMessagezapouzdřují datové struktury MQMD, MQPMO a MQGMO a používají se jako pomůcce při odesílání zpráv do front a načítání zpráv z nich.

Třída MQDistributionList zapouzdřuje kolekci front-local, remote nebo alias pro výstup. Třída položek MQDistributionListzapouzdřuje struktury MQOR, MQRR a MQPMR a přidruží je k seznamu distribučních distribučních položek.

Předání parametru

Parametry při vyvolání metody jsou předávány hodnotou, kromě případů, kdy tento parametr je objekt, a v tom případě je předáván odkaz.

Poskytnutý seznam definic tříd uvádí datový typ pro každý parametr nebo vlastnost. Pro mnoho klientů ActiveX , jako je Visual Basic, pokud použitá proměnná není požadovaného typu, je hodnota automaticky převedena na nebo z požadovaného typu-poskytující takový převod je možný. Tato pravidla se řídí standardními pravidly klienta; MQAX takový převod neposkytuje.

Mnohé z metod používají řetězcové parametry pevné délky nebo vracejí znakový řetězec pevné délky. Převodní pravidla jsou následující:

- Pokud uživatel zadá řetězec s pevnou délkou o chybné délce, jako vstupní parametr nebo jako návratovou hodnotu, hodnota je oseknuta nebo doplněna koncovými mezerami, jak je požadováno.
- Pokud uživatel zadá řetězec s proměnnou délkou o nesprávné délce jako vstupní parametr, je hodnota oříznuta nebo doplněna koncovými mezerami.
- Pokud uživatel zadá řetězec proměnné délky chybné délky jako návratovou hodnotu, řetězec se upraví na požadovanou délku (protože vrací hodnotu, zničí předchozí hodnotu v řetězci i tak).
- Řetězce poskytnuté jako vstupní parametry mohou obsahovat vložené hodnoty Null.

Tyto třídy lze nalézt v knihovně MQAX200 .

Metody přístupu k objektu

Tyto metody se nevztahují přímo k žádnému volání produktu WebSphere MQ . Každá z těchto metod vytvoří objekt, ve kterém jsou umístěny referenční informace, po kterém následuje připojení k objektu WebSphere MQ nebo jeho otevření:

Je-li vytvořeno připojení ke správci front, uchovává atribut 'manipulátor připojení' generovaný produktem WebSphere MQ.

Když je fronta otevřena, uchovává atribut 'object handle' generovaný produktem WebSphere MQ.

Tyto atributy produktu WebSphere MQ nejsou přímo k dispozici pro program MQAX.

Chyby

Syntaktické chyby při předávání parametru mohou být zjištěny v době kompilace a běhové prostředí klientem ActiveX . Chyby mohou být zachyceny pomocí Chyba při chybě ve Visual Basic.

Všechny třídy WebSphere MQ ActiveX obsahují dvě speciální vlastnosti jen pro čtení- ReasonCode a CompletionCode. Tyto vlastnosti lze číst kdykoli.

Pokus o přístup k jakékoli jiné vlastnosti nebo k zadání jakéhokoli volání metody může generovat chybu z produktu WebSphere MQ.

Je-li vlastnost nebo vyvolání metody úspěšné, je hodnota ReasonCode vlastnícího objektu nastavena na hodnotu MQRC_NONE a položka CompletionCode je nastavena na hodnotu MQCC_OK.

Pokud není přístup k vlastnosti nebo vyvolání metody úspěšný, jsou v těchto polích nastaveny kódy příčiny a dokončení.

Třída MQSession

Jedná se o kořenovou třídu pro třídy automatizace produktu WebSphere MQ pro ActiveX.

Pro klientský proces ActiveX vždy existuje pouze jeden objekt MQSession. Při pokusu o vytvoření druhého objektu se vytvoří druhý odkaz na původní objekt.

VYTVOŘENÍ

Volba **Nový** vytvoří nový objekt MQSession.

Syntaxe

Dim *mqsess* **Jako nový MQSession Nastavit** *mqsess* = **Nová relace MQSession**

Vlastnosti

- “Vlastnost CompletionCode” na stránce 1008.
- “Vlastnost ExceptionThreshold” na stránce 1009.
- “Vlastnost ReasonCode” na stránce 1009.
- “Vlastnost ReasonName” na stránce 1009.

Metoda

- “Metoda AccessGetMessageOptions” na stránce 1010.
- “Metoda AccessMessage” na stránce 1010.
- “Metoda AccessPutMessageOptions” na stránce 1010.
- “Metoda AccessQueueManager” na stránce 1010.
- “Metoda ClearErrorCodes” na stránce 1010.
- “Metoda názvu ReasonCode” na stránce 1010.

Vlastnost CompletionCode

Pouze pro čtení. Vrací kód dokončení WebSphere MQ nastavený nejnovější metodou nebo sadou vlastností, která byla vydána na libovolném objektu WebSphere MQ .

Při úspěšném vyvolání metody nebo sady vlastností pro kterýkoli objekt MQAX je obnovení metody MQCC_OK resetováno.

Obslužná rutina události chyby může zkontrolovat tuto vlastnost a diagnostikovat chybu, aniž by bylo nutné vědět, který objekt byl zahrnut.

Použití CompletionCode a ReasonCode v objektu MQSession je velmi výhodné pro jednoduché obslužné rutiny chyb.

Poznámka: Omezení týkající se použití chybových kódů MQSession v aplikacích s podporou podprocesů naleznete v příručce “Využití vláken” na stránce 1005 .

Definováno v: Třída MQSession

Datový typ: Long

Hodnoty:

- MQCC_OK
- VAROVÁNÍ MQCC_WARNING
- SELHÁNÍ MQCC_FAILED

Syntaxe:

Chcete-li získat: *completioncode & = MQSession*.CompletionCode

Vlastnost ExceptionThreshold

Čtení a zápis. Definuje úroveň chyby produktu WebSphere MQ , pro kterou aplikace MQAX vygeneruje výjimku. Výchozí hodnota je MQCC_FAILED. Hodnota větší než MQCC_FAILED účinně zabraňuje zpracování výjimek, takže programátor může provést kontroly na CompletionCode a ReasonCode.

Definováno v: Třída MQSession

Datový typ: Long

Hodnoty:

- Jakýkoli, ale zvažte MQCC_WARNING, MQCC_FAILED nebo vyšší.

Syntaxe:

Chcete-li získat následující informace: *ExceptionThreshold& = MQSession*.**ExceptionThreshold**

Pro nastavení: *MQSession*.**ExceptionThreshold** = *ExceptionThreshold$*

Vlastnost ReasonCode

Pouze pro čtení. Vrátí kód příčiny nastavený nejnovější metodou nebo sadou vlastností, která byla vydána na libovolném objektu WebSphere MQ .

Obslužná rutina události chyby může zkontrolovat tuto vlastnost a diagnostikovat chybu, aniž by bylo nutné vědět, který objekt byl zahrnut.

Použití CompletionCode a ReasonCode v objektu MQSession je velmi výhodné pro jednoduché obslužné rutiny chyb.

Poznámka: Omezení týkající se použití chybových kódů MQSession v aplikacích s podporou podprocesů naleznete v příručce “Využití vláken” na stránce 1005 .

Definováno v: Třída MQSession

Datový typ: Long

Hodnoty:

- Viz Příčina (MQLONG) a další hodnoty MQAX uvedené v části “Kódy příčin” na stránce 1073.

Syntaxe: Chcete-li získat: *reasoncode & = MQSession*.**ReasonCode**

Vlastnost ReasonName

Pouze pro čtení. Vrátí symbolický název posledního kódu příčiny. Příklad: "MQRC_QMGR_NOT_AVAILABLE".

Poznámka: Omezení týkající se použití chybových kódů MQSession v aplikacích s podporou podprocesů naleznete v příručce “Využití vláken” na stránce 1005 .

Definováno v: Třída MQSession

Datový typ: Řetězec

Hodnoty:

- Viz termín kód příčiny rozhraní API.

Syntaxe: Chcete-li získat: *reasonname $= MQSession*.**ReasonName**

Metoda AccessGetMessageOptions

Vytvoří nový objekt voleb MQGetMessage.

Definováno v: Třída MQSession

Syntaxe: *gmo = MQSession*.**AccessGetMessageOptions()**

Metoda AccessMessage

Vytvoří nový objekt MQMessage.

Definováno v: Třída MQSession

Syntaxe: *msg = MQSession*.**AccessMessage()**

Metoda AccessPutMessageOptions

Vytvoří nový objekt voleb MQPutMessage.

Definováno v: Třída MQSession

Syntaxe: *pmo = MQSession*.**AccessPutMessageOptions()**

Metoda AccessQueueManager

Vytvoří nový objekt MQQueueManager a připojí jej ke skutečnému správci front prostřednictvím klienta WebSphere MQ MQI nebo serveru WebSphere MQ . Kromě provádění připojení tato metoda také provádí otevření pro objekt správce front.

Je-li v systému instalován klient WebSphere MQ MQI i server WebSphere MQ , budou aplikace MQAX při výchozím nastavení spuštěny proti serveru. Chcete-li spustit produkt MQAX pro klienta, musí být v proměnné prostředí `GMQ_MQ_LIB` určena knihovna vazeb klienta, například nastavte `GMQ_MQ_LIB=mqic.dll`.

V případě instalace pouze klienta není nutné nastavit proměnnou prostředí `GMQ_MQ_LIB` . Není-li tato proměnná nastavena, produkt WebSphere MQ se pokusí načíst soubor amqzst.dll. Pokud tato knihovna DLL není přítomna (jak je tomu v případě instalace klienta), produkt WebSphere MQ se pokusí zavést soubor mqic.dll.

Je-li funkce úspěšná, nastaví hodnotu ConnectionStatus produktu MQQueueManagerna hodnotu TRUE.

Správce front může být připojen k nejvýše jednomu objektu MQQueueManager na instanci ActiveX .

Dojde-li k selhání připojení ke správci front, dojde k vyvolání chybové události a k nastavení objektu MQSession ReasonCode a CompletionCode .

Definováno v: Třída MQSession

Syntaxe: *set qm = MQSession*.**AccessQueueManager** (*Name$*)

Parametr:*Název$* Řetězec. Název správce front, ke kterému má být připojen.

Metoda ClearErrorCodes

Resetuje parametr CompletionCode na MQCC_OK a ReasonCode na MQRC_NONE.

Definováno v: Třída MQSession

Syntaxe: Volejte funkci *MQSession*.**ClearErrorCodes ()**

Metoda názvu ReasonCode

Vrátí název kódu příčiny s danou číselnou hodnotou. Je užitečné poskytnout jasnější indikaci chybových stavů uživatelům. Název je stále poněkud zašifrováno (například ReasonCodeName (2059) je **MQRC_Q_MGR_NOT_AVAILABLE**), takže případné chyby by měly být zachyceny a nahrazeny popisným textem odpovídajícím aplikaci.

Definováno v: Třída MQSession

Syntaxe: *errname $= MQSession*.**ReasonCodeName**(*reasonCode&*)

Parametr:*reasoncode &* Long. Kód příčiny, pro který je vyžadován symbolický název.

Třída MQQueueManager

Tato třída představuje připojení ke správci front. Správce front může být spuštěn lokálně (server WebSphere MQ) nebo vzdáleně s přístupem poskytovaného klientem WebSphere MQ . Aplikace musí vytvořit objekt této třídy a připojit jej ke správci front. Je-li objekt z této třídy zničen, je automaticky odpojen od správce front.

Obsažení

Objekty třídy MQQueue jsou přidruženy k této třídě.

Nový vytvoří nový objekt MQQueueManager a nastaví všechny vlastnosti na počáteční hodnoty. Alternativně použijte metodu AccessQueueManager třídy MQSession.

VYTVOŘENÍ

Nový vytvoří objekt **nový** MQQueueManager a nastaví všechny vlastnosti na počáteční hodnoty. Alternativně použijte metodu AccessQueueManager třídy MQSession.

Syntaxe

Dim *mgr* **As New MQQueueManager set** *mgr* = **New MQQueueManager**

Vlastnosti

- “Vlastnost ID AlternateUser” na stránce 1012.
- “Vlastnost AuthorityEvent” na stránce 1013.
- “Vlastnost BeginOptions” na stránce 1013.
- “Vlastnost definice ChannelAutoDefinition” na stránce 1013.
- “Vlastnost ChannelAutoDefinitionEvent” na stránce 1013.
- “Vlastnost ChannelAutoDefinitionExit” na stránce 1014.
- “Vlastnost CharacterSet” na stránce 1014.
- “Vlastnost CloseOptions” na stránce 1014.
- “Vlastnost CommandInputvlastnostQueueName” na stránce 1014.
- “Vlastnost CommandLevel” na stránce 1014.
- “Vlastnost CompletionCode” na stránce 1015.
- “Vlastnost ConnectionHandle” na stránce 1015.
- “Vlastnost ConnectionStatus” na stránce 1015.
- “Vlastnost ConnectOptions” na stránce 1015.
- “Vlastnost DeadLetterQueueName” na stránce 1016.
- “Vlastnost DefaultTransmissionQueueName” na stránce 1016.
- “Vlastnost popisu” na stránce 1016.
- “Vlastnost DistributionLists” na stránce 1016.
- “Vlastnost InhibitEvent” na stránce 1016.
- “Vlastnost IsConnected” na stránce 1016.
- “Vlastnost IsOpen” na stránce 1017.
- “Vlastnost LocalEvent” na stránce 1017.

- “Vlastnost MaximumHandles” na stránce 1017.
- “Vlastnost MaximumMessageLength” na stránce 1017.
- “Vlastnost MaximumPriority” na stránce 1017.
- “Vlastnost zpráv MaximumUncommitted” na stránce 1018.
- “Vlastnost name” na stránce 1018.
- “Vlastnost ObjectHandle” na stránce 1018.
- “Vlastnost PerformanceEvent” na stránce 1018.
- “Vlastnost platformy” na stránce 1018.
- “Vlastnost ReasonCode” na stránce 1019.
- “Vlastnost ReasonName” na stránce 1019.
- “Vlastnost RemoteEvent” na stránce 1019.
- “Vlastnost události StartStop” na stránce 1019.
- “Vlastnost Dostupnost SyncPoint” na stránce 1019.
- “Vlastnost TriggerInterval” na stránce 1020.

Metody

- “Metoda AccessQueue” na stránce 1020.
- “Metoda seznamu AddDistribution” na stránce 1020.
- “Metoda vrácení” na stránce 1021.
- “Metoda Begin” na stránce 1021.
- “Metoda ClearErrorCodes” na stránce 1021.
- “Metoda Commit” na stránce 1021.
- “Metoda Connect” na stránce 1021.
- “Metoda Disconnect” na stránce 1021.

Přístup k majetku

K následujícím vlastnostem lze přistupovat kdykoli.

- “Vlastnost ID AlternateUser” na stránce 1012.
- “Vlastnost CompletionCode” na stránce 1015.
- “Vlastnost ConnectionStatus” na stránce 1015.
- “Vlastnost ReasonCode” na stránce 1019.

Ke zbývajícím vlastnostem lze přistupovat pouze v případě, že je objekt připojen ke správci front, a ID uživatele je autorizováno pro zjišťování proti tomuto správci front. Je-li nastaveno alternativní ID uživatele a aktuální ID uživatele je oprávněno používat, alternativní ID uživatele se kontroluje místo toho autorizace pro dotaz.

Pokud tyto podmínky neplatí, produkt WebSphere MQ Automation Classes for ActiveX se pokusí připojit ke správci front a automaticky jej otevřít pro zjišťování. Není-li tento výsledek úspěšný, volání nastaví CompletionCode funkce MQCC_FAILED a jeden z následujících kódů příčiny ReasonCodes:

- PORCC_CONNECTION_CONNECTION_LO
- AUTORIZOVANÝ MQRC_NOT_AUTHORIZED
- CHYBA MQRC_Q_MGR_NAME_ERROR
- MQRC_Q_MGR_NOT_AVAILABLE

Vlastnost ID AlternateUser

Čtení a zápis. Alternativní ID uživatele, které má být použito pro ověření přístupu k atributům správce front.

Tato vlastnost nesmí být nastavena, je-li hodnota IsConnected nastavena na hodnotu TRUE.

Tuto vlastnost nelze nastavit, když je objekt otevřený.

Třída **Defined in:** MQQueueManager

Data Type: Řetězec o délce 12 znaků

Syntax: Chcete-li získat: *altuser $= MQQueueManager*.**AlternateUserId** K nastavení: *MQQueueManager*.**AlternateUserId** = *altuser $*

Vlastnost AuthorityEvent

Pouze pro čtení. Atribut AuthorityEvent rozhraní MQI.

Definováno v:
Třída MQQueueManager

Datový typ:
Dlouhý

Hodnoty:

- MQEV_DISABLED
- POVOLENÝ MQEVR_

Syntaxe: Chcete-li získat: *authevent = MQQueueManager*.**AuthorityEvent**

Vlastnost BeginOptions

Čtení a zápis. Jedná se o volby, které se použijí na metodu Begin. Původně MQBO_NONE.

Definováno v:
Třída MQQueueManager

Datový typ:
Dlouhý

Hodnoty:

- MQBO_NONE

Syntaxe: Chcete-li získat: *beginoptions & =MQQueueManager*.**BeginOptions**

Pro nastavení: *MQQueueManager*.**BeginOptions**=*beginoptions &*

Vlastnost definice ChannelAutoDefinition

Pouze pro čtení. Tento ovládací prvek určuje, zda je povolena automatická definice kanálu.

Definováno v:
Třída MQQueueManager

Datový typ:
Dlouhý

Hodnoty:

- MQCHAD_DISABLED
- MQCHAD_ENABLED

Syntaxe: Chcete-li získat: *channelautodef & = MQQueueManager*.**ChannelAutoDefinition**

Vlastnost ChannelAutoDefinitionEvent

Pouze pro čtení. Tento ovládací prvek určuje, zda jsou generovány události s automatickou definicí kanálu.

Definováno v:
Třída MQQueueManager

Datový typ:
Dlouhý

Hodnoty:

- MQEV_DISABLED
- POVOLENÝ MQEVR_

Syntaxe: Chcete-li získat: *channelautodefevent & =MQQueueManager*.**ChannelAutoDefinitionEvent**

Vlastnost ChannelAutoDefinitionExit

Pouze pro čtení. Název uživatelské procedury použité pro automatickou definici kanálu.

Definováno v:
Třída MQQueueManager

Datový typ:
Řetězec

Syntaxe: Chcete-li získat: *channelaautodefexit$= MQQueueManager*.**ChannelAutoDefinitionExit**

Vlastnost CharacterSet

Pouze pro čtení. Atribut CodedCharSetId rozhraní MQI.

Definováno v: třídě MQQueueManager

Datový typ: Long

Syntaxe: Chcete-li získat: *characterset & = MQQueueManager*.**CharacterSet**

Vlastnost CloseOptions

Čtení a zápis. Volby používané k řízení toho, co se stane, když je správce front uzavřen. Počáteční hodnota je MQCO_NONE.

Definováno v:
Třída MQQueueManager

Datový typ:
Dlouhý

Hodnoty:

- MQCO_NONE

Syntaxe: Chcete-li získat následující informace: *closeopt & = MQQueueManager*.**CloseOptions**

Pro nastavení: *MQQueueManager*.**CloseOptions** =*closeopt &*

Vlastnost CommandInputvlastnostQueueName

Pouze pro čtení. Atribut QName CommandInputMQI.

Definováno v: třídě MQQueueManager

Typ dat: Řetězec 48 znaků

Syntaxe: Chcete-li získat: *commandinputqname $= MQQueueManager*.**CommandInputQueueName**

Vlastnost CommandLevel

Pouze pro čtení. Vrací verzi a úroveň implementace správce front WebSphere MQ (atribut MQI CommandLevel)

Definováno v: třídě MQQueueManager

Datový typ: Long

Syntaxe: Chcete-li získat: *level & = MQQueueManager*.**CommandLevel**

Vlastnost CompletionCode

Pouze pro čtení. Vrací kód dokončení nastavený podle poslední metody nebo přístupu k vlastnosti, který byl vydán proti objektu.

Definováno v: třídě MQQueueManager

Datový typ: Long

Hodnoty:

- MQCC_OK
- VAROVÁNÍ MQCC_WARNING
- SELHÁNÍ MQCC_FAILED

Syntaxe: Chcete-li získat: *completioncode & = MQQueueManager*.**CompletionCode**

Vlastnost ConnectionHandle

Pouze pro čtení. Popisovač připojení pro objekt správce front WebSphere MQ .

Definováno v:
Třída MQQueueManager

Datový typ:
Dlouhý

Syntaxe: Chcete-li získat následující informace: *hconn & = MQQueueManager.***ConnectionHandle**

Vlastnost ConnectionStatus

Pouze pro čtení. Indikuje, zda je objekt připojen ke svému správci front, nebo ne.

Definováno v: třídě MQQueueManager

Typ dat: Boolean

Hodnoty:

- TRUE (-1)
- FALSE (0)

Syntaxe: Chcete-li získat: *status = MQQueueManager*.**ConnectionStatus**

Vlastnost ConnectOptions

Čtení a zápis. Tyto volby se týkají metody Connect. Zpočátku MQCNO_NONE.

Definováno v:
Třída MQQueueManager

Datový typ:
Dlouhý

Hodnoty:

- VAZBA MQCNO_STANDARD_BINDING
- VAZBA MQCNO_FASTPATH_BINDING
- MQCNO_NONE

Syntaxe: Chcete-li získat: *connecttoptions & =MQQueueManager*.**ConnectOptions**

Pro nastavení: *MQQueueManager*.**ConnectOptions**=*connecttoptions &*

Vlastnost DeadLetterQueueName

Pouze pro čtení. Atribut QName DeadLetterrozhraní MQI.

Definováno v: třídě MQQueueManager

Typ dat: Řetězec 48 znaků

Syntaxe: pro získání: *dlqname $= MQQueueManager*.**DeadLetterQueueName**

Vlastnost DefaultTransmissionQueueName

Pouze pro čtení. Atribut QName DefXmitMQI.

Definováno v: třídě MQQueueManager

Typ dat: Řetězec 48 znaků

Syntaxe: Chcete-li získat: *defxmitqname $= MQQueueManager*.**DefaultTransmissionQueueName**

Vlastnost popisu

Pouze pro čtení. Atribut QMgrDesc rozhraní MQI.

Definováno v: třídě MQQueueManager

Typ dat: Řetězec o délce 64 znaků

Syntaxe: Chcete-li získat: *description $= MQQueueManager*.**Popis**

Vlastnost DistributionLists

Pouze pro čtení. Jedná se o schopnost správce front podporovat distribuční seznamy.

Definováno v:
Třída MQQueueManager

Datový typ:
Logická hodnota

Hodnoty:

- TRUE (-1)
- FALSE (0)

Syntaxe: Chcete-li získat: *distributionlists= MQQueueManager*.**DistributionLists**

Vlastnost InhibitEvent

Pouze pro čtení. Atribut MQI InhibitEvent .

Definováno v: třídě MQQueueManager

Datový typ: Long

Hodnoty:

- MQEV_DISABLED
- POVOLENÝ MQEVR_

Syntaxe: Chcete-li získat: *inhibevent & = MQQueueManager*.**InhibitEvent**

Vlastnost IsConnected

Hodnota, která označuje, zda je správce front momentálně připojen.

Pouze pro čtení.

Definováno v: třídě MQQueueManager

Typ dat: Boolean

Hodnoty:

- TRUE (-1)
- FALSE (0)

Syntaxe: Chcete-li získat: *isconnected = MQQueueManager*.**IsConnected**

Vlastnost IsOpen

Hodnota, která označuje, zda je správce front aktuálně otevřen pro zjišťování.

Pouze pro čtení.

Definováno v:
Třída MQQueueManager

Datový typ:
Logická hodnota

Hodnoty:

- TRUE (-1)
- FALSE (0)

Syntaxe: Chcete-li získat: *IsOpen = MQQueueManager*.**IsOpen**

Vlastnost LocalEvent

Pouze pro čtení. Atribut LocalEvent rozhraní MQI.

Definováno v: třídě MQQueueManager

Datový typ: Long

Hodnoty:

- MQEV_DISABLED
- POVOLENÝ MQEVR_

Syntaxe: Chcete-li získat: *localevent & = MQQueueManager*.**LocalEvent**

Vlastnost MaximumHandles

Pouze pro čtení. Atribut MaxHandles MQI.

Definováno v: třídě MQQueueManager

Datový typ: Long

Syntaxe: Chcete-li získat: *maxhandles & = MQQueueManager*.**MaximumHandles**

Vlastnost MaximumMessageLength

Pouze pro čtení. Atribut MaxMsgQueue Manager MQI.

Definováno v: třídě MQQueueManager

Datový typ: Long

Syntaxe: Chcete-li získat následující informace: *maxmessagelength & = MQQueueManager*.**MaximumMessageLength**

Vlastnost MaximumPriority

Pouze pro čtení. Atribut MaxPriority rozhraní MQI.

Definováno v: třídě MQQueueManager

Datový typ: Long

Syntaxe: Chcete-li získat: *maxpriority & = MQQueueManager*.**MaximumPriority**

Vlastnost zpráv MaximumUncommitted

Pouze pro čtení. Atribut MaxUncommittedzpráv rozhraní MQI.

Definováno v: třídě MQQueueManager

Datový typ: Long

Syntaxe: Chcete-li získat následující informace: *maxuncommitted & = MQQueueManager*.**MaximumUncommittedMessages**

Vlastnost name

Čtení a zápis. Atribut QMgrName rozhraní MQI (MQI). Tuto vlastnost nelze zapsat, jakmile je připojen objekt MQQueueManager .

Definováno v: třídě MQQueueManager

Typ dat: Řetězec 48 znaků

Syntaxe: Chcete-li získat: *name $= MQQueueManager*.**name**

Chcete-li nastavit: *MQQueueManager*.**name** = *název $*

Poznámka: Visual Basic rezervuje vlastnost "Name" pro použití ve vizuálním rozhraní. Proto při použití ve Visual Basic use lower-case, tj. "name".

Vlastnost ObjectHandle

Pouze pro čtení. Popisovač objektu pro objekt správce front WebSphere MQ .

Definováno v:
Třída MQQueueManager

Datový typ
Dlouhý

Syntaxe: Chcete-li získat: *hobj & = MQQueueManager.***ObjectHandle**

Vlastnost PerformanceEvent

Pouze pro čtení. Atribut PerformanceEvent rozhraní MQI.

Definováno v: třídě MQQueueManager

Datový typ: Long

Hodnoty:

- MQEV_DISABLED
- POVOLENÝ MQEVR_

Syntaxe: Chcete-li získat: *perfevent & = MQQueueManager*.PerformanceEvent

Vlastnost platformy

Pouze pro čtení. Atribut platformy MQI.

Definováno v: třídě MQQueueManager

Datový typ: Long

Hodnoty:

- POČ MQPL_WINDOWS_NT
- OKNA MQPL_WINDOWS

Syntaxe: Chcete-li získat: *platform & = MQQueueManager*.**Platforma**

Vlastnost ReasonCode

Pouze pro čtení. Vrátí kód příčiny nastavený posledním voláním metody nebo vlastností, které byly vydány pro daný objekt.

Definováno v: třídě MQQueueManager

Datový typ: Long

Hodnoty:

- Viz termín kód příčiny rozhraní API.

Syntaxe: Chcete-li získat následující informace: *reasoncode & = MQQueueManager*.**ReasonCode**

Vlastnost ReasonName

Pouze pro čtení. Vrátí symbolický název posledního kódu příčiny. Příklad: "MQRC_QMGR_NOT_AVAILABLE".

Definováno v: třídě MQQueueManager

Datový typ: Řetězec

Hodnoty:

- Viz termín kód příčiny rozhraní API.

Syntaxe: Chcete-li získat následující informace: *reasonname $= MQQueueManager*.**ReasonName**

Vlastnost RemoteEvent

Pouze pro čtení. Atribut RemoteEvent rozhraní MQI.

Definováno v: třídě MQQueueManager

Datový typ: Long

Hodnoty:

- MQEV_DISABLED
- POVOLENÝ MQEVR_

Syntaxe: Chcete-li získat: *remoteevent & = MQQueueManager*.**RemoteEvent**

Vlastnost události StartStop

Pouze pro čtení. Atribut události rozhraní MQI StartStop.

Definováno v: třídě MQQueueManager

Datový typ: Long

Hodnoty:

- MQEV_DISABLED
- POVOLENÝ MQEVR_

Syntaxe: Chcete-li získat: *strstpevent & = MQQueueManager*.**StartStopEvent**

Vlastnost Dostupnost SyncPoint

Pouze pro čtení. Atribut SyncPoint rozhraní MQI.

Definováno v: třídě MQQueueManager

Datový typ: Long

Hodnoty:

- MQSP_AVAILABLE

- MQSP_NOT_AVAILABLE

Syntaxe: Chcete-li získat: *syncpointailability & = MQQueueManager*.**SyncPointAvailability**

Vlastnost TriggerInterval

Pouze pro čtení. Atribut TriggerInterval MQI.

Definováno v: třídě MQQueueManager

Datový typ: Long

Syntaxe: Chcete-li získat: *trigint & = MQQueueManager*.**TriggerInterval**

Metoda AccessQueue

Vytvoří objekt MQQueue a přidruží jej k tomuto objektu MQQueueManager nastavením vlastnosti odkazu na připojení fronty. Nastavuje vlastnosti Název, OpenOptions, DynamicQueuea AlternateUserID objektu MQQueue na zadané hodnoty a pokusí se ji otevřít.

Je-li otevření neúspěšné, volání selže. U objektu je vyvolána chybová událost. Jsou nastaveny volby ReasonCode a CompletionCodea MQSession ReasonCode a CompletionCode objektu.

Parametry DynamicQueue, QueueManagera AlternateUserID jsou volitelné a výchozí hodnoty "".

Pokud mají být načteny vlastnosti fronty, měly by být kromě jiných voleb určeny také parametry OpenOption MQO_INQUIRE.

Nenastavujte název objektu QueueManagernebo jej nastavte na hodnotu "", je-li fronta, která má být otevřena, lokální. Jinak jej nastavte na název vzdáleného správce front, který je vlastníkem fronty, a dojde k pokusu o otevření lokální definice vzdálené fronty. Další informace o rozlišování názvů vzdálených front a aliasing správce front naleznete v části Co jsou aliasy? .

Je-li vlastnost Název nastavena na název modelové fronty, zadejte název dynamické fronty, která má být vytvořena, v parametru DynamicQueueName$. Je-li hodnota zadaná v parametru DynamicQueueName$ nastavena na hodnotu "", hodnota nastavená na objekt fronty a použitá v otevřeném volání je "AMQ. *". Další informace o pojmenovávání dynamických front naleznete v příručce “Vytváření dynamických front” na stránce 215 .

Definice

Definováno v: Třída MQQueueManager .

Syntaxe

Syntaxe: set queue = MQQueueManager.**AccessQueue**(Name$, OpenOptions&, QueueManagerName$, DynamicQueueName$, AlternateUserId$)

Parametry

Název$ Řetězec. Název fronty produktu WebSphere MQ .

OpenOptions: Long. Volby, které mají být použity při otevření fronty. Viz OpenOptions (MQLONG).

Řetězec *QueueManagerName$* . Název správce front, který vlastní frontu, jež má být otevřena. Hodnota "" znamená, že správce front je lokální.

Řetězec *DynamicQueueName$* . Název přiřazený k dynamické frontě v době, kdy je fronta otevřena, když parametr Name$ uvádí modelovou frontu.

AlternateUserId$ String. Alternativní ID uživatele použité k ověření přístupu při otevření fronty.

Metoda seznamu AddDistribution

Vytvoří nový objekt MQDistributionList a nastaví svůj odkaz na připojení na vlastníka správce front.

Definováno v:
Třída MQQueueManager

Syntaxe: *set distributtionlist* = **MQQueueManager.***AddDistributionList*

Metoda vrácení

Zálohuje všechny nepotvrzené zprávy typu put a gets, které se vyskytly jako část transakce od posledního synchronizačního bodu.

Definováno v: třídě MQQueueManager

Syntaxe: Volejte *MQQueueManager.***Backout ()**

Metoda Begin

Začne jednotku práce, která je koordinována správcem front. Volby zahájení ovlivňují chování této metody.

Definováno v:
Třída MQQueueManager

Syntaxe: Volejte funkci *MQQueueManager*.**Begin ()**

Metoda ClearErrorCodes

Resetuje parametr CompletionCode na MQCC_OK a ReasonCode na MQRC_NONE jak pro třídu MQQueueManager , tak pro třídu MQSession.

Definováno v: třídě MQQueueManager

Syntaxe: Call *MQQueueManager***.ClearErrorCodes ()**

Metoda Commit

Potvrzuje všechny zprávy put a gets, které se vyskytly jako součást pracovní jednotky od posledního synchronizačního bodu.

Definováno v: třídě MQQueueManager

Syntaxe: Volejte *MQQueueManager***.Potvrdit ()**

Metoda Connect

Připojí objekt MQQueueManager ke skutečnému správci front prostřednictvím klienta nebo serveru WebSphere MQ MQI. Stejně jako při vytváření připojení tato metoda také otevře objekt správce front, aby mohl být dotazován.

Nastaví IsConnected na TRUE.

Pro připojení ke správci front je povoleno maximálně jeden objekt MQQueueManager pro instanci ActiveX .

Definováno v: třídě MQQueueManager

Syntaxe: Volejte *MQQueueManager***.Připojit ()**

Metoda Disconnect

Odpojí objekt MQQueueManager od správce front.

Nastaví hodnotu IsConnected na hodnotu FALSE.

Všechny objekty fronty přidružené k objektu MQQueueManager jsou nepoužitelné a nelze je znovu otevřít.

Veškeré nepotvrzené změny (operace vložení a získávání zpráv) jsou potvrzeny.

Definováno v: třídě MQQueueManager

Syntaxe: Volejte *MQQueueManager***.Odpojit ()**

Třída MQQueue

Tato třída představuje přístup k frontě produktu WebSphere MQ . Toto připojení je poskytováno přidruženým objektem MQQueueManager . Je-li objekt z této třídy zničen, automaticky se zavře.

Obsažení

Třída MQQueue je obsažena ve třídě MQQueueManager .

VYTVOŘENÍ

Produkt New vytvoří nový objekt MQQueue a nastaví všechny vlastnosti na počáteční hodnoty. Případně můžete použít metodu AccessQueue třídy MQQueueManager .

Syntaxe

```
Dim que As New MQQueue Set que  = New MQQueue
```

Vlastnosti

- “Vlastnost ID AlternateUser” na stránce 1024.
- “Vlastnost názvu BackoutRequeue” na stránce 1024.
- “Vlastnost BackoutThreshold” na stránce 1025.
- “Vlastnost názvu BaseQueue” na stránce 1025.
- “Vlastnost CloseOptions” na stránce 1025.
- “Vlastnost CompletionCode” na stránce 1025.
- “Vlastnost ConnectionReference” na stránce 1026.
- “Vlastnost Time CreationDate” na stránce 1026.
- “Vlastnost CurrentDepth” na stránce 1026.
- “Vlastnost DefaultInputOpenOption” na stránce 1026.
- “Vlastnost DefaultPersistence” na stránce 1026.
- “Vlastnost DefaultPriority” na stránce 1026.
- “Vlastnost DefinitionType” na stránce 1027.
- “DepthHigh-vlastnost události” na stránce 1027.
- “DepthHighOmezit vlastnost” na stránce 1027.
- “DepthLow-vlastnost události” na stránce 1027.
- “DepthLow-vlastnosti omezení” na stránce 1027.
- “DepthMaximumVlastnost události” na stránce 1027.
- “DepthHigh-vlastnost události” na stránce 1027.
- “DepthHighOmezit vlastnost” na stránce 1027.
- “DepthLow-vlastnost události” na stránce 1027.
- “DepthLow-vlastnosti omezení” na stránce 1027.
- “DepthMaximumVlastnost události” na stránce 1027.
- “Vlastnost popisu” na stránce 1028.
- “Vlastnost názvu DynamicQueue” na stránce 1028.
- “Vlastnost HardenGetBackout” na stránce 1028.
- “Vlastnost InhibitGet” na stránce 1028.
- “Vlastnost InhibitPut” na stránce 1029.

- “Vlastnost názvu InitiationQueue” na stránce 1029.
- “Vlastnost IsOpen” na stránce 1029.
- “Vlastnost MaximumDepth” na stránce 1029.
- “Vlastnost MaximumMessageLength” na stránce 1029.
- “Vlastnost posloupnosti MessageDelivery” na stránce 1029.
- “Vlastnost ObjectHandle” na stránce 1030.
- “Vlastnost OpenInputCount” na stránce 1030.
- “Vlastnost OpenOptions” na stránce 1030.
- “Vlastnost počtu OpenOutput” na stránce 1030.
- “Vlastnost OpenStatus” na stránce 1031.
- “Vlastnost ProcessName” na stránce 1031.
- “Vlastnost názvu QueueManager” na stránce 1031.
- “Vlastnost QueueType” na stránce 1031.
- “Vlastnost ReasonCode” na stránce 1031.
- “Vlastnost ReasonName” na stránce 1032.
- “Vlastnost RemoteQueueManagerName” na stránce 1032.
- “Vlastnost názvu RemoteQueueName” na stránce 1032.
- “Vlastnost ResolvedQueueManagerName” na stránce 1032.
- “Vlastnost názvu ResolvedQueue” na stránce 1032.
- “Vlastnost RetentionInterval” na stránce 1032.
- “Vlastnost scope” na stránce 1032.
- “Vlastnost ServiceInterval” na stránce 1033.
- “Vlastnost události ServiceInterval” na stránce 1033.
- “Vlastnost Shareability” na stránce 1033.
- “Vlastnost názvu TransmissionQueue” na stránce 1033.
- “Vlastnost TriggerControl” na stránce 1033.
- “Vlastnost TriggerData” na stránce 1034.
- “Vlastnost TriggerDepth” na stránce 1034.
- “Vlastnost priority TriggerMessage” na stránce 1034.
- “Vlastnost TriggerType” na stránce 1034.
- “vlastnost použití” na stránce 1034.

Metody

- “Metoda ClearErrorCodes” na stránce 1035
- “Metoda Close” na stránce 1035
- “metoda GET” na stránce 1035
- “Otevřít metodu” na stránce 1036
- “metoda PUT” na stránce 1036

Přístup k majetku

Není-li objekt fronty připojen ke správci front, můžete si přečíst následující vlastnosti:

- “Vlastnost CompletionCode” na stránce 1025
- “Vlastnost OpenStatus” na stránce 1031

- “Vlastnost ReasonCode” na stránce 1031

a můžete číst a zapisovat do:

- “Vlastnost ID AlternateUser” na stránce 1024
- “Vlastnost CloseOptions” na stránce 1025
- “Vlastnost ConnectionReference” na stránce 1026
- “Vlastnost name” na stránce 1030
- “Vlastnost OpenOptions” na stránce 1030

Je-li objekt fronty připojen ke správci front, můžete si přečíst všechny vlastnosti.

Vlastnosti atributu fronty

Vlastnosti, které nejsou uvedeny v předchozí sekci, jsou všechny atributy základní fronty produktu WebSphere MQ . K nim lze přistupovat pouze tehdy, je-li objekt připojen ke správci front a jméno uživatele uživatele je autorizováno pro zjišťování nebo sadu proti této frontě. Je-li nastaveno alternativní ID uživatele a aktuální ID uživatele je pro jeho použití autorizováno, ID alternativního uživatele se kontroluje místo toho oprávnění.

Vlastnost musí být vhodnou vlastností pro daný typ QueueType. Další informace naleznete v tématu Atributy pro fronty .

Pokud tyto podmínky neplatí, přístup k vlastnosti nastaví parametr CompletionCode MQCC_FAILED a jeden z následujících ReasonCodes:

- PORCC_CONNECTION_CONNECTION_LO
- AUTORIZOVANÝ MQRC_NOT_AUTHORIZED
- CHYBA MQRC_Q_MGR_NAME_ERROR
- PŘIPOJENÍ MQRC_Q_MGR_NOT_CONNECTED
- MQRC_SELECTOR_NOT_FOR_TYPE (CompletionCode je MQCC_WARNING)

Otevření fronty

Jediným způsobem, jak vytvořit objekt MQQueue, je použití metody AccessQueue MQQueueManager nebo pomocí volby Nový. Otevřený objekt MQQueue zůstane otevřený (OpenStatus= TRUE), dokud nebude odstraněn nebo odstraněn, nebo dokud nebude objekt správce front odstraněn nebo pokud se ztratí připojení ke správci front. Hodnota vlastnosti MQQueue CloseOptions řídí chování operace zavření, ke které dojde, když je objekt MQQueue odstraněn.

Metoda MQQueueManager AccessQueue otevírá frontu pomocí parametru OpenOptions . Metoda MQQueue.Open otevře frontu s použitím vlastnosti OpenOptions . Produkt WebSphere MQ ověří platnost OpenOptions proti autorizaci uživatele jako součásti procesu otevřené fronty.

Vlastnost ID AlternateUser

Čtení a zápis. Alternativní ID uživatele použité pro ověření přístupu k frontě při jeho otevření.

Tuto vlastnost nelze nastavit při otevření objektu (to znamená, že pokud je položka IsOpen TRUE).

Definováno v: Třída MQQueue

Datový typ: Řetězec o délce 12 znaků

Syntaxe: Chcete-li získat: *altuser $= MQQueue*.**AlternateUserId**

Pro nastavení: *MQQueue.***AlternateUserId** = *altuser $*

Vlastnost názvu BackoutRequeue

Pouze pro čtení. Atribut BackOutRequeueQName MQI.

Definováno v: Třída MQQueue

Typ dat: Řetězec 48 znaků

Syntaxe: Chcete-li získat následující informace: *backoutrequeuename $= MQQueue*.**BackoutRequeueName**

Vlastnost BackoutThreshold

Pouze pro čtení. Atribut BackoutThreshold rozhraní MQI.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- Viz BackoutThreshold (MQLONG).

Syntaxe: Chcete-li získat: *backoutthreshold & = MQQueue.***BackoutThreshold**

Vlastnost názvu BaseQueue

Pouze pro čtení. Název fronty, na kterou je alias vyřešen.

Platné pouze pro alias fronty.

Definováno v: Třída MQQueue

Typ dat: Řetězec 48 znaků

Syntaxe: Chcete-li získat: *baseqname $= MQQueue*.**BaseQueueName**

Vlastnost CloseOptions

Čtení a zápis. Volby používané k řízení toho, co se stane, když je fronta zavřena.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- MQCO_NONE
- MQCO_DELETE
- MQCO_DELETE_PURGE

MQCO_DELETE a MQCO_DELETE_PURGE jsou platné pouze pro dynamické fronty.

Syntaxe: Chcete-li získat: *closeopt & = MQQueue*.**CloseOptions**

Chcete-li nastavit: *MQQueue*.**CloseOptions** = *closeopt &*

Vlastnost CompletionCode

Pouze pro čtení. Vrací kód dokončení nastavený podle poslední metody nebo přístupu k vlastnosti, který byl vydán proti objektu.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- MQCC_OK
- VAROVÁNÍ MQCC_WARNING
- SELHÁNÍ MQCC_FAILED

Syntaxe: Chcete-li získat: *completioncode & = MQQueue*.**CompletionCode**

Vlastnost ConnectionReference

Čtení a zápis. Definuje objekt správce front, do kterého náleží objekt fronty. Odkaz na připojení nelze zapsat, když je fronta otevřená.

Definováno v: Třída MQQueue

Datový typ: MQQueueManager

Hodnoty:

- Odkaz na aktivní objekt produktu WebSphere MQ Queue Manager

Syntaxe: Chcete-li nastavit: *set MQQueue***.ConnectionReference** = *ConnectionReference*

Chcete-li získat následující informace: *set ConnectionReference = MQQueue***.ConnectionReference**

Vlastnost Time CreationDate

Pouze pro čtení. Datum a čas vytvoření této fronty.

Definováno v: Třída MQQueue

Typ dat: Varianta typu 7 (datum/čas) nebo EMPTY

Syntaxe: Chcete-li získat: *datetime = MQQueue***.CreationDateTime**

Vlastnost CurrentDepth

Pouze pro čtení. Počet zpráv, které se právě nacházejí ve frontě.

Definováno v: Třída MQQueue

Datový typ: Long

Syntaxe: Chcete-li získat: *currentdepth & = MQQueue***.CurrentDepth**

Vlastnost DefaultInputOpenOption

Pouze pro čtení. Řídí způsob, jakým je fronta otevřena, pokud OpenOptions uvádí MQOO_INPUT_AS_Q_DEF.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- MQO_INPUT_EXCLUSIVE
- MQO_INPUT_SHARED

Syntaxe: Chcete-li získat: *defaultinop & = MQQueue***.DefaultInputOpenOption**

Vlastnost DefaultPersistence

Pouze pro čtení. Výchozí trvání pro zprávy ve frontě.

Definováno v: Třída MQQueue

Datový typ: Long

Syntaxe: Chcete-li získat: *defpersistence & = MQQueue***.DefaultPersistence**

Vlastnost DefaultPriority

Pouze pro čtení. Výchozí priorita pro zprávy ve frontě.

Definováno v: Třída MQQueue

Datový typ: Long

Syntaxe: Chcete-li získat: *defPriority & = MQQueue*.**DefaultPriority**

Vlastnost DefinitionType

Pouze pro čtení. Typ definice fronty.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- MQQDT_PREDEFINED
- MQQDT_PERMANENT_DYNAMIC
- MQQDT_DOČASNÝ_DYNAMICKÝ

Syntaxe: Chcete-li získat: *deftype & = MQQueue*.**DefinitionType**

DepthHigh-vlastnost události

Pouze pro čtení. Atribut události rozhraní MQI QDepthHigh.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- MQEV_DISABLED
- POVOLENÝ MQEVR_

Syntaxe: Chcete-li získat: *depthhighevent & = MQQueue.*.**DepthHighEvent**

DepthHighOmezit vlastnost

Pouze pro čtení. Atribut Omezení QDepthHighMQI.

Definováno v: Třída MQQueue

Datový typ: Long

Syntaxe: Chcete-li získat: *depthhighlimit & = MQQueue.*.**DepthHighLimit**

DepthLow-vlastnost události

Pouze pro čtení. Atribut události rozhraní MQI QDepthLow.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- MQEV_DISABLED
- POVOLENÝ MQEVR_

Syntaxe: Chcete-li získat: *depthlowevent & = MQQueue.*.**DepthLowEvent**

DepthLow-vlastnosti omezení

Pouze pro čtení. Atribut Omezení QDepthLowMQI.

Definováno v: Třída MQQueue

Datový typ: Long

Syntaxe: Chcete-li získat: *depthlowlimit & = MQQueue.*.**DepthLowLimit**

DepthMaximumVlastnost události

Pouze pro čtení. Atribut události rozhraní MQI QDepthMax.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- MQEV_DISABLED
- POVOLENÝ MQEVR_

Syntaxe: Chcete-li získat: *depthmaximumevent & = MQQueue.***DepthMaximumEvent**

Vlastnost popisu

Pouze pro čtení. Popis fronty.

Definováno v: Třída MQQueue

Typ dat: Řetězec o délce 64 znaků

Syntaxe: Chcete-li získat: *description $= MQQueue***.Popis**

Vlastnost názvu DynamicQueue

Čtení a zápis, pouze pro čtení, je-li fronta otevřená.

Tento parametr řídí název dynamické fronty použité při otevření modelové fronty. Může být nastaven jako zástupný znak uživatelem jako sada vlastností (pouze v případě, že je fronta uzavřena) nebo jako parametr pro MQQueueManager.AccessQueue().

Skutečný název dynamické fronty se nachází dotazem QueueName.

Definováno v: Třída MQQueue

Typ dat: Řetězec 48 znaků

Hodnoty:

- Jakýkoli platný název fronty produktu WebSphere MQ .

Syntaxe: Chcete-li nastavit: *MQQueue***.DynamicQueueName** = *dynamickqueuename $*

Chcete-li získat: *dynamicqueuename $ = MQQueue***.DynamicQueueName**

Vlastnost HardenGetBackout

Pouze pro čtení. Zda se má udržovat přesný zpětný počet.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- MACKY_BACKOUT_HARDENED
- MQQA_BACKOUT_NOT HARDENED

Syntaxe: Chcete-li získat: *hardengetback & = MQQueue***.HardenGetBackout**

Vlastnost InhibitGet

Čtení a zápis. Atribut InhibitGet rozhraní MQI.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- MQQA_GET_INHIBED

- MQQA_GET_ALLOWED

Syntaxe: Chcete-li získat: *getstatus & = MQQueue*.**InhibitGet**

Pro nastavení: *MQQueue*.**InhibitGet** = *getstatus &*

Vlastnost InhibitPut

Čtení a zápis. Atribut MQI InhibitPut .

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- MQQA_PUT_BLOKOVÁNO
- MQQA_PUT_ALLOWED

Syntaxe: Chcete-li získat: *putstatus & = MQQueue*.**InhibitPut**

Pro nastavení: *MQQueue*.**InhibitPut** = *putstatus &*

Vlastnost názvu InitiationQueue

Pouze pro čtení. Název inicializační fronty.

Definováno v: Třída MQQueue

Typ dat: Řetězec 48 znaků

Syntaxe: Chcete-li získat: *initqname $= MQQueue*.**InitiationQueueName**

Vlastnost IsOpen

Vrátí informaci o tom, zda je fronta otevřená.

Pouze pro čtení.

Definováno v: Třída MQQueue

Typ dat: Boolean

Hodnoty:

- TRUE (-1)
- FALSE (0)

Syntaxe: Chcete-li získat: *open = MQQueue*.**IsOpen**

Vlastnost MaximumDepth

Pouze pro čtení. Maximální hloubka fronty.

Definováno v: Třída MQQueue

Datový typ: Long

Syntaxe: Chcete-li získat: *maxdepth & = MQQueue*.**MaximumDepth**

Vlastnost MaximumMessageLength

Pouze pro čtení. Maximální povolená délka zprávy v bajtech pro tuto frontu.

Definováno v: Třída MQQueue

Datový typ: Long

Syntaxe: Chcete-li získat: *maxmlength & = MQQueue*.**MaximumMessageLength**

Vlastnost posloupnosti MessageDelivery

Pouze pro čtení. Sekvence doručení zpráv.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- PRIORITA MQMS_PRIORITY
- MQSD_FIFO

Syntaxe: Chcete-li získat: *messagedeseq & = MQQueue*.**MessageDeliverySequence**

Vlastnost name

Čtení a zápis. Atribut Fronta MQI. Tuto vlastnost nelze zapsat poté, co je otevřena fronta MQQueue.

Definováno v: Třída MQQueue

Typ dat: Řetězec 48 znaků

Syntaxe: Chcete-li získat: *name $= MQQueue*.**name**

Chcete-li nastavit: *MQQueue*.**name** = *název $*

Poznámka: Visual Basic rezervuje vlastnost "Name" pro použití ve vizuálním rozhraní. Proto při použití ve Visual Basic use lower-case, to je "name".

Vlastnost ObjectHandle

Pouze pro čtení. Popisovač objektu pro objekt fronty WebSphere MQ .

Definováno v: Třída MQQueue

Datový typ: Long

Syntaxe: Chcete-li získat: *hobj & = MQQueue.***ObjectHandle**

Vlastnost OpenInputCount

Pouze pro čtení. Počet otevření pro vstup.

Definováno v: Třída MQQueue

Datový typ: Long

Syntaxe: Chcete-li získat: *openincount & = MQQueue*.**OpenInputCount**

Vlastnost OpenOptions

Čtení a zápis. Volby, které mají být použity pro otevření fronty.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- Viz OpenOptions (MQLONG).

Syntaxe: Chcete-li získat: *openopt & = MQQueue*.**OpenOptions**

Pro nastavení: *MQQueue.***OpenOptions** = *openopt &*

Vlastnost počtu OpenOutput

Pouze pro čtení. Počet operací otevření pro výstup.

Definováno v: Třída MQQueue

Datový typ: Long

Syntaxe: Chcete-li získat: *openoutcount & = MQQueue*.**OpenOutputCount**

Vlastnost OpenStatus

Pouze pro čtení. Označuje, zda je fronta otevřena či nikoli. Počáteční hodnota je TRUE po metodě AccessQueue nebo FALSE po Nové.

Definováno v: Třída MQQueue

Typ dat: Boolean

Hodnoty:

- TRUE (-1)
- FALSE (0)

Syntaxe: Chcete-li získat: *status & = MQQueue*.**OpenStatus**

Vlastnost ProcessName

Pouze pro čtení. Atribut MQI ProcessName .

Definováno v: Třída MQQueue

Typ dat: Řetězec 48 znaků

Syntaxe: Chcete-li získat: *procname $ =MQQueue*.**ProcessName**

Vlastnost názvu QueueManager

Čtení a zápis. Název správce front produktu WebSphere MQ .

Definováno v: Třída MQQueue

Datový typ: Řetězec

Syntaxe: Chcete-li získat následující informace: *QueueManagerName$ = MQQueue*.**QueueManagerName**

Chcete-li nastavit: *MQQueue*.**QueueManagerName** = *QueueManagerName$*

Vlastnost QueueType

Pouze pro čtení. Atribut QType rozhraní MQI.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- ALIAS MQQ_ALIAS
- MQQ_LOCAL
- MQQ_MODEL
- MQQT_REMOTE

Syntaxe: Chcete-li získat: *queuetype & = MQQueue*.**QueueType**

Vlastnost ReasonCode

Pouze pro čtení. Vrátí kód příčiny nastavený posledním voláním metody nebo vlastností, které byly vydány pro daný objekt.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- Viz termín kód příčiny rozhraní API.

Syntaxe: Chcete-li získat: *reasoncode & = MQQueue*.**ReasonCode**

Vlastnost ReasonName

Pouze pro čtení. Vrátí symbolický název posledního kódu příčiny. Příklad: "MQRC_QMGR_NOT_AVAILABLE".

Definováno v: Třída MQQueue

Datový typ: Řetězec

Hodnoty:

- Viz termín kód příčiny rozhraní API.

Syntaxe: Chcete-li získat: *reasonname $= MQQueue*.**ReasonName**

Vlastnost RemoteQueueManagerName

Pouze pro čtení. Název vzdáleného správce front. Platí pouze pro vzdálené fronty.

Definováno v: Třída MQQueue

Typ dat: Řetězec 48 znaků

Syntaxe: Chcete-li získat: *remqmanname $= MQQueue*.**RemoteQueueManagerName**

Vlastnost názvu RemoteQueueName

Pouze pro čtení. Název fronty, jak je znám ve vzdáleném správci front. Platí pouze pro vzdálené fronty.

Definováno v: Třída MQQueue

Typ dat: Řetězec 48 znaků

Syntaxe: Chcete-li získat: *remqname $= MQQueue*.**RemoteQueueName**

Vlastnost ResolvedQueueManagerName

Pouze pro čtení. Název konečného cílového správce front, jak je znám pro lokálního správce front.

Definováno v: Třída MQQueue

Typ dat: Řetězec 48 znaků

Syntaxe: Chcete-li získat: *resqmanname $= MQQueue*.**ResolvedQueueManagerName**

Vlastnost názvu ResolvedQueue

Pouze pro čtení. Název konečné cílové fronty, jak je znám pro lokálního správce front.

Definováno v: Třída MQQueue

Typ dat: Řetězec 48 znaků

Syntaxe: Chcete-li získat: *reqname $= MQQueue*.**ResolvedQueueNázev**

Vlastnost RetentionInterval

Pouze pro čtení. Doba, po kterou by měla být fronta zadržena.

Definováno v: Třída MQQueue

Datový typ: Long

Syntaxe: Chcete-li získat: *retinterval & = MQQueue*.**RetentionInterval**

Vlastnost scope

Pouze pro čtení. Určuje, zda položka pro tuto frontu také existuje v adresáři buňky.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- MQSCOM_Q_MGR
- BUŇKA MQSCO_CELL

Syntaxe: Chcete-li získat: *scope & = MQQueue*.**Scope**

Vlastnost ServiceInterval

Pouze pro čtení. Atribut MQI QServiceInterval .

Definováno v: Třída MQQueue

Datový typ: Long

Syntaxe: Pro získání: *serviceinterval & = MQQueue.*.**ServiceInterval**

Vlastnost události ServiceInterval

Pouze pro čtení. Atribut události rozhraní MQI QServiceInterval.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- MQQSIE_HIGH
- MQQSIE_OK
- MQQSIE_NONE

Syntaxe: Chcete-li získat: *serviceintervalevent & = MQQueue.*.**ServiceIntervalEvent**

Vlastnost Shareability

Pouze pro čtení. Možnost sdílení fronty.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- MQQA_SHAREABLE
- MQQA_NOT_SHAREABLE

Syntaxe: Chcete-li získat: *shareability & = MQQueue*.**Shareability**

Vlastnost názvu TransmissionQueue

Pouze pro čtení. Název přenosové fronty. Platí pouze pro vzdálené fronty.

Definováno v: Třída MQQueue

Typ dat: Řetězec 48 znaků

Syntaxe: Chcete-li získat: *transqname $= MQQueue*.**TransmissionQueueName**

Vlastnost TriggerControl

Čtení a zápis. Řízení spouštěče.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- MQTC_OFF
- MQTC_ON

Syntaxe: Chcete-li získat následující informace: *trigcontrol & = MQQueue***.TriggerControl**

Pro nastavení: *MQQueue***.TriggerControl** = *trigcontrol &*

Vlastnost TriggerData

Čtení a zápis. Data spouątěče.

Definováno v: Třída MQQueue

Typ dat: Řetězec o délce 64 znaků

Syntaxe: Chcete-li získat: *trigdata $= MQQueue***.TriggerData**

Chcete-li nastavit: *MQQueue***.TriggerData** = *trigdata $*

Vlastnost TriggerDepth

Čtení a zápis. Počet zpráv, které musí být ve frontě, než je zapsána zpráva spouštěče.

Definováno v: Třída MQQueue

Datový typ: Long

Syntaxe: Chcete-li získat: *trigdepth & = MQQueue***.TriggerDepth**

Pro nastavení: *MQQueue***.TriggerDepth** = *trigdepth &*

Vlastnost priority TriggerMessage

Čtení a zápis. Priorita zprávy prahové hodnoty pro spouątěče.

Definováno v: Třída MQQueue

Datový typ: Long

Syntaxe: Chcete-li získat: *trigmesspriority & = MQQueue***.TriggerMessagePriority**

Pro nastavení: *MQQueue***.TriggerMessagePriority** = *trigmesspriority &*

Vlastnost TriggerType

Čtení a zápis. Typ spouątěče.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- MQTTE_NONE
- NEJPRVE MQTT_FIRST
- MQTT_EVERY
- MQTT_DEPTH

Syntaxe: Chcete-li získat: *trigtype & = MQQueue***.TriggerType**

Chcete-li nastavit: *MQQueue***.TriggerType** = *Trigtype &*

vlastnost použití

Pouze pro čtení. Označuje, pro kterou frontu se používá fronta.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- MQUS_NORMAL
- PŘENOS MQUS_TRANSMISSION

Syntaxe: Chcete-li získat: *usage & = MQQueue*.**Využití**

Metoda ClearErrorCodes

Resetuje CompletionCode na MQCC_OK a ReasonCode na MQRC_NONE jak pro třídu MQQueue, tak pro třídu MQSession.

Definováno v: Třída MQQueue

Syntaxe: Volejte funkci *MQQueue*.**ClearErrorCodes ()**

Metoda Close

Zavře frontu s použitím aktuálních hodnot parametru CloseOptions.

Definováno v: Třída MQQueue

Syntaxe: Volat frontu *MQQueue*.**Zavřít ()**

metoda GET

Načte zprávu z fronty.

Tato metoda vezme objekt MQMessage jako parametr s použitím některých polí v deskriptoru MQMD objektu jako vstupních parametrů. Konkrétně jsou použita pole MessageId a CorrelId , takže je důležité zajistit, aby byla tato pole nastavena podle potřeby. Další informace o těchto polích naleznete v části MsgId (MQBYTE24) a CorrelId (MQBYTE24) .

Pokud metoda selže, objekt MQMessage se nezmění. Pokud uspěje, datové části MQMD a Message Data objektu MQMessage budou nahrazeny daty MQMD a Data zprávy z příchozí zprávy. Vlastnosti řízení MQMessage jsou nastaveny takto:

- **MessageLength** je nastavena na délku zprávy produktu WebSphere MQ .
- **DataLength** je nastaven na délku zprávy WebSphere MQ
- **DataOffset** je nastaven na nulu.

Definováno v:
Třída MQQueue

Syntaxe: Volat frontu *MQQueue*.**Get**(*Message, GetMsgOptions, GetMsgLength*)

Parametry
Zpráva:

Objekt MQMessage Object reprezentující zprávu, která má být načtena.

Volby GetMsg:

Volitelný objekt Volby MQGetMessagepro řízení operace get. Není-li tento parametr zadán, použijí se výchozí volby MQGetMessage.

GetMsgDélka:

Volitelná hodnota délky 2 nebo 4, která slouží k řízení maximální délky zprávy WebSphere MQ načtené z fronty.

Je-li zadána volba MQGMO_ACCEPT_TRUNCATED_MSG, operace GET se úspěšně provede s kódem dokončení MQCC_WARNING a kódem příčiny MQRC_TRUNCATED_MSG_ACCEPTED, pokud velikost zprávy překročí zadanou délku.

Položka MessageData uchovává první GetMsgLength bajtů dat.

Je-li zadána hodnota MQGMO_ACCEPT_TRUNCATED_MSG ***není*** a velikost zprávy překračuje uvedenou délku, je vrácen kód dokončení operace MQCC_FAILED spolu s kódem příčiny MQRC_TRUNCATED_MESSAGE_FAILED.

Není-li definován obsah vyrovnávací paměti zpráv, je celková délka zprávy nastavena na plnou délku zprávy, která by byla načtena.

Není-li parametr délky zprávy zadán, bude délka vyrovnávací paměti zpráv automaticky upravena alespoň na velikost příchozí zprávy.

Otevřít metodu

Otevře frontu s použitím aktuálních hodnot:

1. QueueName
2. QueueManagerName
3. AlternateUserid
4. Název DynamicQueue

Definováno v:
Třída MQQueue

Syntaxe: Volat frontu *MQQueue*.**Otevřít ()**

metoda PUT

Umístí zprávu do fronty.

Tato metoda přijímá objekt MQMessage jako parametr. Vlastnosti objektu MQMD (Message Descriptor) tohoto objektu mohou být v důsledku této metody změněny. Hodnoty, které mají bezprostředně po spuštění této metody, jsou hodnotami, které byly vloženy do produktu WebSphere MQ.

Úpravy objektu MQMessage po dokončení operace Put nemají vliv na skutečnou zprávu ve frontě WebSphere MQ .

Definováno v:
Třída MQQueue

Syntaxe: Volat frontu *MQQueue*.**Vložení**(*Zpráva, volby PutMsg)*

Parametry
Zpráva

Objekt MQMessage reprezentující zprávu, která má být vložena.

Volby PutMsg

Objekt voleb MQPutMessageobsahující volby pro řízení operace vložení. Nejsou-li tyto hodnoty zadány, použijí se výchozí volby PutMessage.

Třída MQMessage

Tato třída představuje zprávu produktu WebSphere MQ . Obsahuje vlastnosti pro zapouzdření deskriptoru zpráv WebSphere MQ (MQMD) a poskytuje vyrovnávací paměť pro uložení dat zpráv definovaných aplikací.

Třída obsahuje metody zápisu pro kopírování dat z aplikace ActiveX do objektu MQMessage. Podobně třída zahrnuje metody čtení pro kopírování dat z objektu MQMessage do aplikace ActiveX . Třída spravuje alokaci a dealokaci paměti pro vyrovnávací paměť automaticky. Aplikace nemusí deklarovat velikost vyrovnávací paměti při vytvoření objektu MQMessage, protože vyrovnávací paměť dorůstá do přijetí dat, která jsou do ní zapsána.

Nemůžete umístit zprávu do fronty WebSphere MQ , pokud velikost vyrovnávací paměti překročí vlastnost MaximumMessageLength dané fronty.

Po vytvoření objektu MQMessage lze objekt MQMessage umístit do fronty produktu WebSphere MQ pomocí metody MQQueue.Put . Tato metoda vezme kopii dat MQMD a datových částí zprávy objektu a umístí tuto kopii do fronty. Aplikace proto může po vložení upravit nebo odstranit objekt MQMessage, aniž by to mělo vliv na zprávu ve frontě WebSphere MQ . Správce front může upravit některá pole v deskriptoru MQMD při kopírování zprávy ve frontě WebSphere MQ .

Příchozí zprávu lze číst do objektu MQMessage pomocí metody MQQueue.Get . Tím se nahradí veškerá data MQMD nebo zprávy, která mohla být v objektu MQMessage s hodnotami z příchozí zprávy, která již byla v objektu MQMessage. Upravuje velikost vyrovnávací paměti dat objektu MQMessage tak, aby odpovídala velikosti příchozích dat zprávy.

Obsažení

Zprávy jsou obsaženy ve třídě MQSession.

VYTVOŘENÍ

Volba **Nový** vytvoří objekt MQMessage. Jeho vlastnosti deskriptoru zpráv jsou na počátku nastaveny na výchozí hodnoty a její vyrovnávací paměť dat je prázdná.

Syntaxe

```
Dim msg As New MQMessage or Set msg = New MQMessage
```

Vlastnosti

Vlastnosti ovládacích prvků jsou:

- “Vlastnost CompletionCode” na stránce 1039
- “Vlastnost DataLength” na stránce 1040
- “Vlastnost DataOffset” na stránce 1040
- “Vlastnost MessageLength” na stránce 1040
- “Vlastnost ReasonCode” na stránce 1040
- “Vlastnost ReasonName” na stránce 1041

Vlastnosti deskriptoru zpráv jsou:

- “Vlastnost AccountingToken” na stránce 1041
- “AccountingTokenHexadecimální vlastnost” na stránce 1041
- “Datová vlastnost ApplicationId” na stránce 1041
- “vlastnost dat ApplicationOrigin” na stránce 1041
- “Vlastnost BackoutCount” na stránce 1042
- “Vlastnost CharacterSet” na stránce 1042
- “Vlastnost CorrelationId” na stránce 1042
- “CorrelationIdHexadecimální vlastnost” na stránce 1043
- “Vlastnost kódování” na stránce 1043
- “Vlastnost vypršení platnosti” na stránce 1044
- “Vlastnost Feedback” na stránce 1044
- “Vlastnost formátu” na stránce 1044
- “Vlastnost GroupId” na stránce 1044
- “GroupIdHexadecimální vlastnost” na stránce 1045
- “Vlastnost MessageData” na stránce 1045

Metody

- “Metoda WriteDecimal4” na stránce 1054
- “Metoda WriteDouble” na stránce 1054
- “Metoda WriteDouble4” na stránce 1055
- “Metoda WriteFloat” na stránce 1055
- “Metoda WriteInt2” na stránce 1055
- “Metoda WriteInt4” na stránce 1055
- “Metoda WriteLong” na stránce 1055
- “Metoda WriteNullTerminatedString” na stránce 1056
- “Metoda WriteShort” na stránce 1056
- “Metoda WriteString” na stránce 1056
- “Metoda WriteUInt2” na stránce 1056
- “WriteUnsignedBytová metoda” na stránce 1057
- “Metoda WriteUTF” na stránce 1057

Přístup k vlastnosti

Všechny vlastnosti lze číst kdykoli.

Řídicí vlastnosti jsou určeny pouze pro čtení, kromě DataOffset , který je čtení-zápis. Vlastnosti deskriptoru zpráv jsou všechny pro čtení-zápis, kromě BackoutCount a TotalMessageLength, které jsou určeny pouze pro čtení.

Mějte však na paměti, že některé vlastnosti MQMD mohou správce front upravit při vložení zprávy do fronty produktu WebSphere MQ . Podrobné informace o tom, jak mohou být upraveny, najdete v polích v MQMD .

Převod dat

Můžete předat binární data do zprávy produktu WebSphere MQ nastavením vlastnosti CharacterSet na identifikátor kódované znakové sady správce front (MQCCSI_Q_MMGR) a předat jej řetězec. Funkci chr $můžete použít k nastavení neznakových dat do řetězce.

Metody čtení a zápisu provádějí převod dat. Konvertují mezi vnitřním formátem ActiveX a formáty zpráv WebSphere MQ , jak jsou definovány vlastnostmi Encoding a CharacterSet z deskriptoru zpráv. Při zápisu zprávy nastavte hodnoty do kódování a CharacterSet , která odpovídá charakteristikám příjemce zprávy před vydáním metody Write. Při čtení zprávy se tento krok obvykle nevyžaduje, protože tyto hodnoty budou nastaveny z hodnot v příchozím MQMD.

Jedná se o další krok konverze dat, který se stane po provedení jakékoli konverze provedené metodou MQQueue.Get .

Vlastnost CompletionCode

Pouze pro čtení. Vrací kód dokončení WebSphere MQ nastavený pomocí poslední metody nebo přístupu k vlastnosti vydaného pro tento objekt.

Definováno v: třídě MQMessage

Datový typ: Long

Hodnoty:

- MQCC_OK
- VAROVÁNÍ MQCC_WARNING
- SELHÁNÍ MQCC_FAILED

Syntaxe: Chcete-li získat: *completioncode & = MQMessage*.**CompletionCode**

Vlastnost DataLength

Pouze pro čtení. Tato vlastnost vrací hodnotu:

```
MQMessage.MessageLength - MQMessage.DataOffset
```

Lze jej použít před metodou čtení, abyste zkontrolli, že očekávaný počet znaků je ve skutečnosti přítomen ve vyrovnávací paměti.

Počáteční hodnota je nula.

Definováno v: třídě MQMessage

Datový typ: Long

Syntaxe: Chcete-li získat: *bytesaleft & = MQMessage***.DataLength**

Vlastnost DataOffset

Čtení a zápis. Aktuální pozice v části Data zprávy objektu zprávy.

Hodnota je vyjádřena jako bajtový posun od začátku vyrovnávací paměti dat zprávy; první znak ve vyrovnávací paměti odpovídá hodnotě DataOffset nulové hodnoty.

Metoda čtení nebo zápisu zahajuje svou činnost na znaku, na který se odkazuje DataOffset. Tyto metody zpracují data ve vyrovnávací paměti sekvenčně z této pozice a aktualizují hodnotu DataOffset tak, aby ukazovala na bajt (je-li nějaký) bezprostředně za posledním zpracovávanými bajty.

Hodnota DataOffset může obsahovat pouze hodnoty v rozsahu od nuly do MessageLength včetně. Když DataOffset = MessageLength ukazuje na konec, to je první neplatný znak vyrovnávací paměti. Metody zápisu jsou v této situaci povoleny-rozšiřují data ve vyrovnávací paměti a zvyšují MessageLength o počtu přidaných bajtů. Čtení nad konec vyrovnávací paměti není platné.

Počáteční hodnota je nula.

Definováno v: třídě MQMessage

Datový typ: Long

Syntaxe: Chcete-li získat: *curpos & = MQMessage***.DataOffset**

Pro nastavení: *MQMessage***.DataOffset** = *curpos &*

Vlastnost MessageLength

Pouze pro čtení. Vrátí celkovou délku části dat zprávy objektu zprávy ve znacích bez ohledu na hodnotu parametru DataOffset.

Počáteční hodnota je nula. Je nastavena na příchozí délku zprávy po vyvolání metody Get, která odkazuje na tento objekt zprávy. Tato hodnota se zvýší, pokud aplikace používá metodu Write k přidání dat do objektu. To není ovlivněno metodami čtení.

Definováno v: třídě MQMessage

Datový typ: Long

Syntaxe: Chcete-li získat: *msglolength & = MQMessage***.MessageLength**

Vlastnost ReasonCode

Pouze pro čtení. Vrátí kód příčiny nastavený nejnovější metodou nebo úrovní přístupu k vlastnosti vydaným pro tento objekt.

Definováno v: třídě MQMessage

Datový typ: Long

Hodnoty:

- Viz termín kód příčiny rozhraní API.

Syntaxe: Chcete-li získat: *reasoncode & = MQMessage*.**ReasonCode**

Vlastnost ReasonName

Pouze pro čtení. Vrátí symbolický název posledního kódu příčiny. Příklad: "MQRC_QMGR_NOT_AVAILABLE". **Definováno v:** třídě MQMessage

Datový typ: Řetězec

Hodnoty:

- Viz termín kód příčiny rozhraní API.

Syntaxe: Chcete-li získat: *reasonname $= MQMessage*.**ReasonName**

Vlastnost AccountingToken

Čtení a zápis. MQMD AccountingToken -část kontextu identity zprávy.

Jeho počáteční hodnota je všechny hodnoty null.

Definováno v: třídě MQMessage

Typ dat: Řetězec o délce 32 znaků

Syntaxe: Chcete-li získat: *actoken $= MQMessage*.**AccountingToken**

Chcete-li nastavit: *MQMessage.***AccountingToken** = *actoken $*

Další informace o tom, kdy musíte použít AccountingTokenHex na místě vlastnosti AccountingToken , naleznete v tématu “Vlastnosti deskriptoru zpráv” na stránce 1001 .

AccountingTokenHexadecimální vlastnost

Čtení a zápis. MQMD AccountingToken -část kontextu identity zprávy.

Každé dva znaky reprezentují hexadecimální ekvivalent jednoho znaku ASCII. Například dvojice znaků "6" a "1" představují jediný znak "A", dvojice znaků "6" a "2" představují jediný znak "B" atd.

Je třeba zadat 64 platných hexadecimálních znaků.

Jeho počáteční hodnota je "0 ... 0"

Definováno v: třídě MQMessage

Typ dat: Řetězec 64 hexadecimálních znaků představujících 32 znaků ASCII

Syntaxe: Chcete-li získat: *actokenh $= MQMessage*.**AccountingTokenHex**

Chcete-li nastavit: *MQMessage.***AccountingTokenHex** = *actokenh $*

Další informace o tom, kdy musíte použít AccountingTokenHex na místě vlastnosti AccountingToken , naleznete v tématu “Vlastnosti deskriptoru zpráv” na stránce 1001 .

Datová vlastnost ApplicationId

Čtení a zápis. Data produktu MQMD ApplIdentityData-Část kontextu identity zprávy.

Jeho počáteční hodnota je prázdná.

Definováno v: třídě MQMessage

Typ dat: Řetězec o délce 32 znaků

Syntaxe: Chcete-li získat: *applid $= MQMessage*.**ApplicationIdData**

Nastavení: *MQMessage*.**ApplicationIdData** = *applid $*

vlastnost dat ApplicationOrigin

Čtení a zápis. MQMD ApplOriginData-Část kontextu zprávy o původu zprávy.

Jeho počáteční hodnota je prázdná.

Definováno v: třídě MQMessage

Typ dat: Řetězec o délce 4 znaků

Syntaxe: Chcete-li získat: *applor $= MQMessage***.ApplicationOriginData**

Chcete-li nastavit: *MQMessage***.ApplicationOriginData** = *applor $*

Vlastnost BackoutCount

Pouze pro čtení. MQMD BackoutCount.

Jeho počáteční hodnota je 0

Definováno v: třídě MQMessage

Datový typ: Long

Syntaxe: Chcete-li získat: *backoutct & = MQMessage***.BackoutCount**

Vlastnost CharacterSet

Čtení a zápis. MQMD CodedCharSetId.

Jeho počáteční hodnota je speciální hodnota **MQCCSI_Q_MGR** .

Je-li vlastnost CharacterSet nastavena na hodnotu **MQCCSI_Q_MGR**, použije se kódová stránka pro aktuální národní prostředí pro znakovou konverzi v metodě WriteString . Pro serverové aplikace je použitá kódová stránka kódovou stránkou správce front. Pro klientské aplikace se jedná o výchozí aktuální kódovou stránku národního prostředí.

Příklad:

```
msg.CharacterSet = MQCCSI_Q_MGR
msg.WriteString(chr$(n))
```

kde 'n' je větší nebo rovno nule a menší nebo rovno 255, vede k tomu, že jediný bajt hodnoty' n' je zapsán do vyrovnávací paměti.

Definováno v: třídě MQMessage

Datový typ: Long

Syntaxe: Chcete-li získat: *:30ccid& = MQMessage* **.CharacterSet**

Pro nastavení: *MQMessage***.CharacterSet**= *ccid &*

Příklad

Pokud chcete řetězec zapsaný v kódové stránce 437, zadejte:

```
Message.CharacterSet = 437
Message.WriteString ("string to be written")
```

Před zadáním jakéhokoli volání WriteString nastavte hodnotu, kterou chcete v souboru CharacterSet .

Vlastnost CorrelationId

Čtení a zápis. Identifikátor CorrelationId , který má být zahrnut do MQMD zprávy při vložení do fronty. Také ID, které se bude porovnávat při získávání zprávy z fronty.

Jeho počáteční hodnota je null.

Definováno v: třídě MQMessage

Typ dat: Řetězec o délce 24 znaků

Syntaxe: Chcete-li získat: *correlid $= MQMessage*.**CorrelationId** To set: *MQMessage*.**CorrelationId** = *correlid $*

Další informace o tom, kdy musíte použít CorrelationIdHex místo vlastnosti CorrelationId , najdete v tématu “Vlastnosti deskriptoru zpráv” na stránce 1001 .

CorrelationIdHexadecimální vlastnost

Čtení a zápis. Identifikátor CorrelationId , který má být zahrnut do MQMD zprávy při vložení do fronty. Také CorrelationId , které se má porovnávat při získávání zprávy z fronty.

Každé dva znaky řetězce reprezentují hexadecimální ekvivalent jednoho znaku ASCII. Například dvojice znaků "6" a "1" představují jediný znak "A", dvojice znaků "6" a "2" představují jediný znak "B" atd.

Je třeba zadat 48 platných hexadecimálních znaků.

Jeho počáteční hodnota je "0 ... 0".

Definováno v: třídě MQMessage

Typ dat: Řetězec 48 hexadecimálních znaků představujících 24 znaků ASCII

Syntaxe: Chcete-li získat: *correlidh $= MQMessage*.**CorrelationIdHex**

Nastavení: *MQMessage*.**CorrelationIdHex** = *correlidh $*

Diskuze o tom, kdy musíte použít CorrelationIdHex místo vlastnosti CorrelationId , najdete v tématu “Vlastnosti deskriptoru zpráv” na stránce 1001 .

Vlastnost kódování

Čtení a zápis. Pole MQMD, které identifikuje znázornění použité pro číselné hodnoty v datech zprávy aplikace.

Jeho počáteční hodnota je speciální hodnota MQENC_NATIVE, která se liší podle platformy.

Tato vlastnost je používána následujícími metodami:

- Metoda ReadDecimal2
- Metoda ReadDecimal4
- Metoda ReadDouble
- Metoda ReadDouble4
- Metoda ReadFloat
- Metoda ReadInt2
- Metoda ReadInt4
- Metoda ReadLong
- Metoda ReadShort
- Metoda ReadUInt2
- Metoda WriteDecimal2
- Metoda WriteDecimal4
- Metoda WriteDouble
- Metoda WriteDouble4
- Metoda WriteFloat
- Metoda WriteInt2
- Metoda WriteInt4
- Metoda WriteLong
- Metoda WriteShort
- Metoda WriteUInt2

Definováno v: třídě MQMessage

Datový typ: Long

Syntaxe: Chcete-li získat: *encoding & = MQMessage*.**Kódování** Chcete-li nastavit: *MQMessage*.**Kódování** = *encoding &*

Pokud připravujete zápis dat do vyrovnávací paměti zpráv, měli byste toto pole nastavit tak, aby odpovídalo charakteristikám přijímající platformy správce front, pokud přijímající správce front není schopen provést vlastní převod dat.

Vlastnost vypršení platnosti

Čtení a zápis. Pole vypršení platnosti MQMD se očekává v desetinách sekundy.

Jeho počáteční hodnota je speciální hodnota MQEI_UNLIMITED.

Definováno v: třídě MQMessage

Datový typ: Long

Syntaxe: Chcete-li získat: *expiry & = MQMessage*.**Vypršení platnosti**

Nastavení: *MQMessage*.**Expiry** = *expiry &*

Vlastnost Feedback

Čtení a zápis. Pole zpětné vazby MQMD.

Jeho počáteční hodnota je speciální hodnota MQFB_NONE.

Definováno v: třídě MQMessage

Datový typ: Long

Hodnoty:

- Viz Váš názor.

Syntaxe: Chcete-li získat: *feedback & = MQMessage*.**Váš názor**

Nastavení: *MQMessage*.**Zpětná vazba** = *feedback &*

Vlastnost formátu

Čtení a zápis. Pole formátu MQMD. Poskytuje název vestavěného nebo uživatelem definovaného formátu, který popisuje povahu dat zprávy.

Jeho počáteční hodnotou je speciální hodnota MQFMT_NONE.

Definováno v: třídě MQMessage

Typ dat: Řetězec o délce 8 znaků

Syntaxe: Chcete-li získat: *format $= MQMessage*.**Formát**

Nastavení: *MQMessage*.**Formát** = *formát $*

Vlastnost GroupId

Čtení a zápis. Hodnota GroupId , která má být zahrnuta do zprávy MQPMR zprávy při vložení do fronty. Také ID, které se bude porovnávat při získávání zprávy z fronty. Jeho počáteční hodnota je všechny hodnoty null.

Definováno v:
Třída MQMessage

Datový typ:
Řetězec o délce 24 znaků

Syntaxe: Chcete-li získat: *groupid $= MQMessage*.**GroupId**

Chcete-li nastavit: *MQMessage*.**GroupId** = *groupid $*

Další informace o tom, kdy je třeba použít GroupIdHex místo vlastnosti GroupId , naleznete v části "Vlastnosti deskriptoru zpráv" na stránce 1001 .

GroupIdHexadecimální vlastnost

Čtení a zápis. Hodnota GroupId , která má být zahrnuta do zprávy MQPMR zprávy při vložení do fronty. Také ID, které se bude porovnávat při získávání zprávy z fronty.

Každé dva znaky řetězce reprezentují hexadecimální ekvivalent jednoho znaku ASCII. Například dvojice znaků "6" a "1" představují jediný znak "A", dvojice znaků "6" a "2" představují jediný znak "B" a tak dále.

Je třeba zadat 48 platných hexadecimálních znaků.

Jeho počáteční hodnota je "0 ... 0".

Definováno v:
Třída MQMessage

Datový typ:
Řetězec 48 hexadecimálních znaků představujících 24 znaků ASCII.

Syntaxe: Chcete-li získat: *groupidh $= MQMessage*.**GroupIdHex**

Chcete-li nastavit: *MQMessage*.**GroupIdHex** = *groupidh $*

Další informace o tom, kdy je třeba použít GroupIdHex místo vlastnosti GroupId , naleznete v části "Vlastnosti deskriptoru zpráv" na stránce 1001 .

Vlastnost MessageData

Čtení a zápis. Načte nebo nastaví celý obsah zprávy jako znakový řetězec.

Definováno v: třídě MQMessage

Datový typ: Varianta

Poznámka: Datový typ používaný touto vlastností je Variant, ale MQAX očekává, že se jedná o typ varianty String. Pokud předáte v jiné variantě než tento typ, bude vrácena chyba MQRC_OBJECT_TYPE_ERROR.

Syntaxe: Chcete-li získat: *String$ = MQMessage***.MessageData**

Pro nastavení: *MQMessage***.MessageData** = *String$*

Vlastnost MessageFlags

Čtení a zápis. Příznaky zprávy určující řídicí informace Segmentace. Počáteční hodnota je 0.

Definováno v:
Třída MQMessage

Datový typ:
Dlouhý

Hodnoty:
Viz MsgFlags (MQLONG).

Syntaxe: Chcete-li získat: *messageflags & = MQMessage*.**MessageFlags**

Pro nastavení: *MQMessage*.**MessageFlags** = *messageflags &*

Vlastnost MessageId

Čtení a zápis. Hodnota MessageId , která má být zahrnuta do MQMD zprávy při vložení do fronty. Také ID, které se bude porovnávat při získávání zprávy z fronty.

Jeho počáteční hodnota je všechny hodnoty null.

Definováno v: třídě MQMessage

Typ dat: Řetězec o délce 24 znaků

Syntaxe: Chcete-li získat: *messageid $= MQMessage*.**MessageId**

Pro nastavení: *MQMessage*.**MessageId** = *messageid $*

Další informace o tom, kdy je třeba použít MessageIdHex místo vlastnosti MessageId , naleznete v tématu “Vlastnosti deskriptoru zpráv” na stránce 1001 .

MessageIdHexadecimální vlastnost

Čtení a zápis. Hodnota MessageId , která má být zahrnuta do MQMD zprávy při vložení do fronty. Také MessageId , které má být porovnávána při získávání zprávy z fronty.

Každé dva znaky řetězce reprezentují hexadecimální ekvivalent jednoho znaku ASCII. Například dvojice znaků "6" a "1" představují jediný znak "A", dvojice znaků "6" a "2" představují jediný znak "B" atd.

Je třeba zadat 48 platných hexadecimálních znaků.

Jeho počáteční hodnota je "0 ... 0".

Definováno v: třídě MQMessage

Typ dat: Řetězec 48 hexadecimálních znaků představujících 24 znaků ASCII

Syntaxe: Chcete-li získat: *messagesh $= MQMessage*.**MessageIdHex**

Nastavení: *MQMessage*.**MessageIdHex** = *messageidh $* .

Další informace o tom, kdy je třeba použít hodnotu MessageIdHex místo vlastnosti MessageId , naleznete v tématu “Vlastnosti deskriptoru zpráv” na stránce 1001 .

Vlastnost čísla MessageSequence

Čtení a zápis. Informace o posloupnosti identifikující zprávu v rámci skupiny. Počáteční hodnota je 1.

Definováno v:
Třída MQMessage

Datový typ:
Dlouhý

Hodnoty:
Viz MsgSeqNumber (MQLONG).

Syntaxe: Chcete-li získat: *sequengenumber & = MQMessage*.**SequenceNumber**

Pro nastavení: *MQMessage*.**SequenceNumber** = *sequencenumber &*

Vlastnost MessageType

Čtení a zápis. Pole MQMD MsgType .

Jeho počáteční hodnota je MQMT_DATAGRAM.

Definováno v: třídě MQMessage

Datový typ: Long

Hodnoty:

- Viz MsgType (MQLONG).

Syntaxe: Chcete-li získat: *msgtype & = MQMessage*.**MessageType**

Pro nastavení: *MQMessage*.**MessageType** = *msgtype &*

Vlastnost Posunutí

Čtení a zápis. Posunutí v segmentované zprávě. Počáteční hodnota je 0.

Definováno v:
Třída MQMessage

Datový typ:
Dlouhý

Hodnoty:
Viz Posunutí (MQLONG).

Syntaxe: Chcete-li získat: *offset & = MQMessage*.**Offset**

Pro nastavení: *MQMessage*.**Offset** = *offset &*

Vlastnost OriginalLength

Čtení a zápis. Původní délka segmentované zprávy. Počáteční hodnota je MQOL_UNDEFINED.

Definováno v:
Třída MQMessage

Datový typ:
Dlouhý

Hodnoty:
Viz OriginalLength (MQLONG).

Syntaxe: Chcete-li získat: *originallength & = MQMessage*.**OriginalLength**

Pro nastavení: *MQMessage*.**OriginalLength** = *originalllength &*

Vlastnost Persistence

Čtení a zápis. Nastavení perzistence zprávy.

Jeho počáteční hodnota je MQPER_PERSISTENCE_AS_Q_DEF.

Definováno v: třídě MQMessage

Datový typ: Long

Syntaxe: Chcete-li získat: *persist & = MQMessage*.**Perzistence**

Nastavení: *MQMessage*.**Persistence** = *persist &*

Vlastnost priority

Čtení a zápis. Priorita zprávy.

Jeho počáteční hodnota je speciální hodnota MQPRI_PRIORITY_AS_Q_DEF

Definováno v: třídě MQMessage

Datový typ: Long

Syntaxe: Chcete-li získat: *priority & = MQMessage*.**Priorita.**

Nastavení: *MQMessage*.**Priorita** = *priorita &*

Vlastnost názvu PutApplication

Čtení a zápis. Název MQMD PutAppl-část kontextu Původ zprávy.

Jeho počáteční hodnota je prázdná.

Definováno v: třídě MQMessage

Typ dat: Řetězec obsahující 28 znaků

Syntaxe: Chcete-li získat: *putapplnm $= MQMessage*.**PutApplicationName**

Nastavení: *MQMessage*.**PutApplicationName** = *putapplnm $*

Vlastnost typu PutApplication

Čtení a zápis. Typ záhlaví MQMD PutAppl-část kontextu Původ zprávy.

Jeho počáteční hodnota je MQAT_NO_CONTEXT

Definováno v: třídě MQMessage

Datový typ: Long

Hodnoty:

- Viz termín PutApplType (MQLONG).

Syntaxe: Chcete-li získat: *putappltp & = MQMessage*.**PutApplicationType**

Nastavení: *MQMessage*.**PutApplicationType** = *putappltp &*

Vlastnost Time PutDate

Čtení/zápis. Tato vlastnost kombinuje pole MQMD PutDate a PutTime . Jedná se o části kontextu Původ zprávy, které označují, kdy byla zpráva vložena.

Rozšíření ActiveX se převádí mezi formátem data/času ActiveX a formáty data a času použité v produktu WebSphere MQ MQMD. Je-li přijata zpráva, která má neplatnou hodnotu PutDate nebo PutTime, pak je vlastnost PutDateTime po metodě get nastavena na EMPTY.

Jeho počáteční hodnota je EMPTY.

Definováno v: třídě MQMessage

Typ dat: Varianta typu 7 (datum/čas) nebo EMPTY.

Syntaxe: Chcete-li získat: *datetime = MQMessage*.**PutDateTime**

Pro nastavení: *MQMessage*.**PutDateTime** = *datetime* .

Vlastnost ReplyToQueueManagerName

Čtení a zápis. Pole QMgr ReplyToMQMD.

Jeho počáteční hodnota je prázdná

Definováno v: třídě MQMessage

Typ dat: Řetězec 48 znaků

Syntaxe: Chcete-li získat následující informace: *replytoqmgr $= MQMessage*.**ReplyToQueueManagerName**

Pro nastavení: *MQMessage*.**ReplyToQueueManagerName** = *replytoqmgr $*

Vlastnost ReplyToQueueName

Čtení a zápis. Pole Q ReplyToMQMD.

Jeho počáteční hodnota je prázdná

Definováno v: třídě MQMessage

Typ dat: Řetězec 48 znaků

Syntaxe: Chcete-li získat následující informace: *replytoq $= MQMessage*.**ReplyToQueueName**

Pro nastavení: *MQMessage*.**ReplyToQueueName** = *replytoq $*

Vlastnost sestavy

Čtení a zápis. Volby sestavy zprávy.

Jeho počáteční hodnota je MQRO_NONE.

Definováno v: třídě MQMessage

Datový typ: Long

Hodnoty:

- Viz Sestava.

Syntaxe: Chcete-li získat: *report & = MQMessage*.**Sestava**

Nastavení: *MQMessage*.**Sestava** = *report &*

Vlastnost TotalMessageLength

Pouze pro čtení. Načte délku poslední zprávy přijaté příkazem MQGET. Pokud nebyla zpráva zkrácena, tato hodnota se rovná hodnotě vlastnosti MessageLength .

Definováno v: třídě MQMessage

Datový typ: Long

Syntaxe: Chcete-li získat: *totalmessagelength & = MQMessage*.**TotalMessageLength**

vlastnost UserId

Čtení a zápis. MQMD UserIdentifier -část kontextu identity zprávy.

Jeho počáteční hodnota je prázdná.

Definováno v: třídě MQMessage

Datový typ: Řetězec o délce 12 znaků

Syntaxe: Chcete-li získat: *userid $= MQMessage*.**UserId**

Pro nastavení: *MQMessage*.**UserId** = *userid $*

Metoda ClearErrorCodes

Resetuje CompletionCode na MQCC_OK a ReasonCode na MQRC_NONE jak pro třídu MQMessage, tak pro třídu MQSession.

Definováno v: třídě MQMessage

Syntaxe: Volat *MQMessage*.**ClearErrorCodes ()**

Metoda ClearMessage

Tato metoda vymaže část vyrovnávací paměti dat objektu MQMessage. Veškerá data zprávy v datové vyrovnávací paměti se ztratí, protože MessageLength, DataLengtha DataOffset jsou všechny nastaveny na nulu.

Část MQMD (Message Descriptor) není ovlivněna; aplikace může vyžadovat úpravu některých z polí MQMD předtím, než bude znovu použit objekt MQMessage. Chcete-li nastavit pole MQMD zpět, použijte volbu Nový pro nahrazení objektu novou instancí.

Definováno v: třídě MQMessage

Syntaxe: Volat *MQMessage*.**ClearMessage()**

Metoda čtení

Čte posloupnost bajtů z vyrovnávací paměti zpráv do bajtového pole. Hodnota DataOffset se inkrementuje a sníží se o délku dat o počet přečtených bajtů.

Definováno v:
Třída MQMessage

Syntaxe: Data = MQMessage.**Čtení**(len &)

Parametry:
len &: Long. Délka dat v bajtech, která se mají přečíst.

Metoda ReadBoolean

Čte 1bajtovou logickou hodnotu z aktuální pozice v vyrovnávací paměti zpráv a vrací 2bajtovou logickou hodnotu TRUE (-1) /FALSE (0). Hodnota DataOffset je zvýšena o jedničku a délka dat se sníží o jednu.

Definováno v:
Třída MQMessage

Syntaxe: *value = MQMessage*.**ReadBoolean**

Metoda ReadByte

Tato metoda čte 1 bajt z vyrovnávací paměti dat zprávy, počínaje znakem, na který se odkazuje DataOffset a vrací jej jako celé číslo typu Integer (signed 2-byte) v rozsahu -128 až 127.

Metoda selže, pokud je MQMessage.DataLength menší než 1, když je vydána.

Hodnota DataOffset se zvýší o 1 a hodnota DataLength se sníží o 1, je-li metoda úspěšná.

Bytem dat zprávy se předpokládá, že se jedná o podepsané binární celé číslo.

Definováno v: třídě MQMessage

Syntaxe: *intedgerv% = MQMessage***.ReadByte**

Metoda ReadDecimal2

Přečte 2bajtové pakované desetinné číslo a vrátí jej jako podepsanou 2bajtovou celočíselnou hodnotu. Hodnota DataOffset je zvýšena o dvě a délka dat se sníží o dvě.

Definováno v:
Třída MQMessage

Syntaxe: *value% = MQMessage*.**ReadDecimal2**

Metoda ReadDecimal4

Čte 4bajtové pakované desetinné číslo a vrátí ji jako 4bajtovou celočíselnou hodnotu se znaménkem. Hodnota DataOffset se zvýší o čtyři a Data Length se sníží o čtyři.

Definováno v:
Třída MQMessage

Syntaxe: Volání *value & = MQMessage*.**ReadDecimal4**

Metoda ReadDouble

Tato metoda čte z vyrovnávací paměti dat zprávy 8 bajtů, počínaje bajtem, na který se odkazuje hodnota DataOffset a vrací ji jako hodnotu s plovoucí řádovou čárkou ve tvaru Double (se znaménkem 8 bajtů).

Metoda selže, pokud je MQMessage.DataLength menší než 8, když je vydána.

Hodnota DataOffset se zvýší o 8 a hodnota DataLength se sníží o 8, je-li metoda úspěšná.

Předpokládá se, že 8 znaků dat zprávy je binární číslo s pohyblivou řádovou čárkou. Kódování je určeno vlastností MQMessage.Encoding . Všimněte si, že převod z formátu System/360 není podporován.

Definováno v: třídě MQMessage

Syntaxe: *doublev# = MQMessage***.ReadDouble**

Metoda ReadDouble4

Metody ReadDouble4 a WriteDouble4 jsou alternativy k ReadFloat a WriteFloat. Důvodem je to, že podporují 4bajtové hodnoty zpráv s pohyblivou řádovou čárkou System/390 , které jsou příliš velké pro převod na 4bajtový formát IEEE s plovoucí řádovou čárkou.

Tato metoda čte 4 bajty z vyrovnávací paměti dat zprávy, počínaje bajtem, na který odkazuje hodnota DataOffset a vrací ji jako hodnotu s plovoucí řádovou čárkou (s přesností na 8 bajtů).

Metoda selže, pokud je MQMessage.DataLength menší než 4, když je vydána.

Hodnota DataOffset se zvýší o 4 a hodnota DataLength se sníží o 4, je-li metoda úspěšná.

Předpokládá se, že 4 znaky dat zprávy jsou binární číslo s pohyblivou řádovou čárkou. Kódování je určeno vlastností MQMessage.Encoding . Všimněte si, že převod z formátu System/360 není podporován.

Definováno v: třídě MQMessage

Syntaxe: *doublev# = MQMessage.***ReadDouble4**

Metoda ReadFloat

Tato metoda čte 4 bajty z vyrovnávací paměti dat zprávy, počínaje bajtem, na který se odkazuje DataOffset a vrací ji jako hodnotu s plovoucí řádovou čárkou Single (podepsaná 4 bajty).

Metoda selže, pokud je MQMessage.DataLength menší než 4, když je vydána.

Hodnota DataOffset se zvýší o 4 a hodnota DataLength se sníží o 4, je-li metoda úspěšná.

Předpokládá se, že 4 znaky dat zprávy jsou číslo s pohyblivou řádovou čárkou. Kódování je určeno vlastností MQMessage.Encoding . Všimněte si, že převod z formátu System/360 není podporován.

Definováno v: třídě MQMessage

Syntaxe: *singlev! = MQMessage***.ReadFloat**

Metoda ReadInt2

Metoda je identická s metodou ReadShort .

Syntaxe: *intedgerv% = MQMessage***.ReadInt2**

Metoda ReadInt4

Tato metoda je identická s metodou ReadLong .

Syntaxe: *bigint & = MQMessage***.ReadInt4**

Metoda ReadLong

Tato metoda čte 4 bajty z vyrovnávací paměti dat zprávy, počínaje bajtem, na který se odkazuje DataOffset a vrací jej jako hodnotu typu Long (se znaménkem 4-byte).

Metoda selže, pokud je MQMessage.DataLength menší než 4, když je vydána.

Hodnota DataOffset se zvýší o 4 a hodnota DataLength se sníží o 4, je-li metoda úspěšná.

Předpokládá se, že 4 znaky dat zprávy jsou binární celé číslo. Kódování je určeno vlastností MQMessage.Encoding .

Definováno v: třídě MQMessage

Syntaxe: *bigint & = MQMessage***.ReadLong**

Metoda ReadNullTerminatedString

Tato metoda je určena pro použití v místě ReadString , pokud řetězec může obsahovat vložené znaky null.

Tato metoda přečte uvedený počet bajtů z vyrovnávací paměti dat zprávy začínající na bajt, na který odkazuje hodnota DataOffset a vrací jej jako řetězec ActiveX . Pokud řetězec obsahuje vložený znak null před koncem, pak je délka vráceného řetězce redukována tak, aby odrážela pouze ty znaky před hodnotou null.

Hodnota DataOffset je inkrementována a hodnota DataLength se sníží o zadanou hodnotu bez ohledu na to, zda řetězec obsahuje vložené nulové znaky.

Předpokládá se, že znaky v datech zprávy jsou řetězcem v kódové stránce určené vlastností MQMessage.CharacterSet . Konverze na znázornění ActiveX se provádí pro aplikaci.

Definováno v:
Třída MQMessage

Syntaxe: *string $ = MQMessage*.**ReadNullTerminatedString(délka &)**

Parametry:
délka & *Long.* Délka pole řetězce v bajtech.

Metoda ReadShort

Tato metoda čte 2 bajty z vyrovnávací paměti dat zprávy, počínaje bajtem, na který se odkazuje DataOffset a vrací jej jako hodnotu typu Integer (podepsaná 2-byte).

Metoda selže, pokud je MQMessage.DataLength menší než 2, když je vydána.

Hodnota DataOffset se zvýší o 2 a hodnota DataLength se sníží o 2, pokud je metoda úspěšná.

Předpokládá se, že 2 znaky dat zprávy jsou binární celé číslo. Kódování je určeno vlastností MQMessage.Encoding .

Definováno v: třídě MQMessage

Syntaxe: *integerv% = MQMessage*.**ReadShort**

Metoda ReadString

Tato metoda čte n bajtů z vyrovnávací paměti dat zprávy začínající na bajt, na který odkazuje DataOffset a vrací jej jako řetězec ActiveX .

Metoda selže, pokud je MQMessage.DataLength menší než n, když je vydán.

Hodnota DataOffset je zvýšena o n a hodnota DataLength se sníží o n, pokud metoda uspěje.

Předpokládá se, že n znaků dat zprávy je řetězec v kódové stránce určené vlastností MQMessage.CharacterSet . Konverze na znázornění ActiveX se provádí pro aplikaci.

Definováno v: třídě MQMessage

Syntaxe: *stringv $= MQMessage*.**ReadString***(délka &*)

Parametr
délka & Long. Délka pole řetězce v bajtech.

Metoda ReadUInt2

Tato metoda čte 2 bajty z vyrovnávací paměti dat zprávy, počínaje bajtem, na který se odkazuje DataOffset a vrací jej jako hodnotu typu Long (se znaménkem 4-byte).

Metoda selže, pokud je MQMessage.DataLength menší než 2, když je vydána.

Hodnota DataOffset se zvýší o 2 a hodnota DataLength se sníží o 2, pokud je metoda úspěšná.

Předpokládá se, že 2 bajty dat zprávy jsou binární celé číslo bez znaménka. Kódování je určeno vlastností MQMessage.Encoding .

Definováno v: třídě MQMessage

Syntaxe: *bigint & = MQMessage*.**ReadUInt2**

Metoda ReadUnsignedByte

Tato metoda čte 1 bajt z vyrovnávací paměti dat zprávy, počínaje bajtem, na který se odkazuje DataOffset a vrací ji jako celočíselnou hodnotu typu Integer (podepsaná 2-bajt) v rozsahu od 0 do 255.

Metoda selže, pokud je MQMessage.DataLength menší než 1, když je vydána.

Hodnota DataOffset se zvýší o 1 a hodnota DataLength se sníží o 1, je-li metoda úspěšná.

Předpokládá se, že 1 znak dat zprávy je binární celé číslo bez znaménka.

Definováno v: třídě MQMessage

Syntaxe: *integerv% = MQMessage*.**ReadUnsignedByte**

Metoda ReadUTF

Tato metoda čte řetězec formátu UTF ze zprávy začínající na bajt, na který odkazuje hodnota DataOffset , a vrací jej jako řetězec ActiveX . Řetězec ve zprávě se skládá z 2bajtové délky následované znakovými daty.

Metoda selže, pokud je MQMessage.DataLength menší než délka řetězce, když je vydána.

Hodnota DataOffset je zvětšena o délku řetězce a DataLength se sníží o délku řetězce, pokud metoda uspěje.

Definováno v:
Třída MQMessage

Syntaxe: *value $= MQMessage*.**ReadUTF**

Metoda ResizeBuffer

Tato metoda pozmění velikost úložiště, které je momentálně přiděleno interně, aby zadržoval vyrovnávací paměť dat zprávy. Poskytuje aplikaci určitou kontrolu nad automatickou správou vyrovnávací paměti tím, že v případě, že aplikace ví, že se bude zabývat velkou zprávou, může zajistit, že je alokována dostatečně velká vyrovnávací paměť. Aplikace nevyžaduje použití tohoto volání-pokud tomu tak není, bude velikost vyrovnávací paměti zvětšovat velikost vyrovnávací paměti pro automatickou správu vyrovnávací paměti.

Pokud změníte velikost vyrovnávací paměti tak, aby byla menší než aktuální hodnota MessageLength, riskujete ztrátu dat. Pokud ztratíte data, metoda vrací CompletionCode MQCC_WARNING a ReasonCode objektu MQRC_DATA_TRUNCATED.

Pokud změníte velikost vyrovnávací paměti tak, aby byla menší než hodnota vlastnosti **DataOffset** , postupujte takto:

- Vlastnost **DataOffset** se změní tak, aby ukazovala na konec nové vyrovnávací paměti.
- Vlastnost **DataLength** je nastavena na nulu.
- Vlastnost **MessageLength** se změní na novou velikost vyrovnávací paměti.

Definováno v:
Třída MQMessage

Syntaxe: *MQMessage*.**ResizeBuffer**(*Length &*)

Parametr:
Délka & Dlouhé. Velikost je povinná ve znacích.

Metoda zápisu

Zapisuje posloupnost bajtů do vyrovnávací paměti zpráv z bajtového pole na pozici, na kterou se odkazuje posunutí dat. Je-li to nutné, je délka vyrovnávací paměti (MQMessage.MQMessageLength) rozšířena tak, aby obsásla celou délku pole bajtů. Hodnota DataOffset je zvýšena o počet bajtů zapsaných, pokud byla metoda úspěšná.

Definováno v:
Třída MQMessage

Syntaxe: Volejte *MQMessage*.**Write**(hodnota)

Parametry:
data: bajtové pole nebo varianta odkazu na bajtové pole

Metoda WriteBoolean

Zapisuje 1bajtovou logickou hodnotu na aktuální pozici ve vyrovnávací paměti zpráv z 2bajtové logické hodnoty. Hodnota DataOffset se zvýší o jednu.

Definováno v:
Třída MQMessage

Syntaxe: Volejte *MQMessage*.**WriteBoolean**(hodnota)

Parametr:
hodnota: Boolean (2-bajty). Hodnota, která má být zapsána.

Metoda WriteByte

Tato metoda vezme podepsanou 2bajtovou celočíselnou hodnotu a zapíše ji do vyrovnávací paměti dat zprávy jako jednobajtové binární číslo na pozici, na kterou se odkazuje DataOffset. Pokud je to nutné, nahradí data již na pozici ve vyrovnávací paměti a rozšíří délku vyrovnávací paměti (MQMessage.MessageLength).

Hodnota DataOffset se zvýší o jednu, pokud je metoda úspěšná.

Uvedená hodnota by měla být v rozsahu -128 až 127. Pokud tomu tak není, vrátí se metoda CompletionCode MQCC_FAILED a ReasonCode MQRC_WRITE_VALUE_ERROR.

Definováno v: třídě MQMessage

Syntaxe: Volejte *MQMessage***.WriteByte**(*value%*)

Parametr: *value%* Integer. Hodnota, která má být zapsána.

Metoda WriteDecimal2

Zapisuje se 2 bajtové celé číslo jako 2bajtové dekadické číslo se znaménkem. Hodnota DataOffset je zvýšena o dvě.

Definováno v:
Třída MQMessage

Syntaxe: Volejte *MQMessage*.**WriteDecimal2**(hodnota%)

Parametr:
hodnota% celé číslo. Hodnota, která má být zapsána.

Metoda WriteDecimal4

Zapisuje se 4bajtové celé číslo se znaménkem jako 4bajtové dekadické číslo. Hodnota DataOffset je zvýšena o čtyři.

Definováno v:
Třída MQMessage

Syntaxe: Volejte *MQMessage*.**WriteedDecimal4**(value &)

Parametr:
hodnota & Long. Hodnota, která má být zapsána.

Metoda WriteDouble

Tato metoda vezme hodnotu 8bajtového čísla s pohyblivou řádovou čárkou a zapíše ji do vyrovnávací paměti dat zprávy jako 8bajtové číslo s pohyblivou řádovou čárkou, které začíná na pozici, na kterou se odkazuje hodnota DataOffset. Nahrazuje veškerá data již na těchto pozicích ve vyrovnávací paměti a v případě potřeby rozšiřuje délku vyrovnávací paměti (MQMessage.MessageLength).

Hodnota DataOffset je zvýšena o 8, je-li metoda úspěšná.

Metoda se převede na reprezentaci plovoucí řádové čárky zadané vlastností MQMessage.Encoding . *Konverze do formátu System/360 není podporována.*

Definováno v: třídě MQMessage

Syntaxe: Volat *MQMessage*.**WriteDouble**(*value#*)

Parametr:
Hodnota typu Double. Hodnota, která má být zapsána.

Metoda WriteDouble4

Popis nastavení příkazu ReadDouble4 a WriteDouble4 v místě ReadFloat a WriteFloatviz “Metoda ReadDouble4” na stránce 1050 .

Tato metoda vezme hodnotu 8bajtového čísla s pohyblivou řádovou čárkou a zapíše ji do vyrovnávací paměti dat zprávy jako 4bajtové číslo s pohyblivou řádovou čárkou počínaje pozicí, na kterou se odkazuje hodnota DataOffset.

Hodnota DataOffset je zvýšena o 4, je-li metoda úspěšná.

Nahrazuje veškerá data již na těchto pozicích ve vyrovnávací paměti a v případě potřeby rozšiřuje délku vyrovnávací paměti (MQMessage.MessageLength).

Metoda se převede na reprezentaci plovoucí řádové čárky zadané vlastností MQMessage.Encoding . *Konverze do formátu System/360 není podporována.*

Definováno v: třídě MQMessage

Syntaxe: Volat *MQMessage*.**WriteDouble4 (value#)**

Parametr: *value#* Double. Hodnota, která má být zapsána.

Metoda WriteFloat

Tato metoda vezme hodnotu 4 bajtového čísla s pohyblivou řádovou čárkou a zapíše ji do vyrovnávací paměti dat zprávy jako 4bajtové číslo s pohyblivou řádovou čárkou, které začíná na znaku, na který se odkazuje hodnota DataOffset. Nahrazuje veškerá data již na těchto pozicích ve vyrovnávací paměti a v případě potřeby rozšiřuje délku vyrovnávací paměti (MQMessage.MessageLength).

Hodnota DataOffset je zvýšena o 4, je-li metoda úspěšná.

Metoda se převede do binární reprezentace zadané vlastností MQMessage.Encoding . *Konverze do formátu System/360 není podporována.*

Definováno v: třídě MQMessage

Syntaxe: Volejte *MQMessage*.**WriteFloat**(*value!*)

Parametr *hodnota!* Float. Hodnota, která má být zapsána.

Metoda WriteInt2

Tato metoda je identická s metodou WriteShort .

Syntaxe: Volat *MQMessage*.**WriteInt2**(*hodnota%*)

Parametr *hodnota%* celé číslo. Hodnota, která má být zapsána.

Metoda WriteInt4

Tato metoda je identická s metodou WriteLong .

Syntaxe: Volat *MQMessage*.**WriteInt4**(*hodnota &*)

Parametr *hodnota &* Long. Hodnota, která má být zapsána.

Metoda WriteLong

Tato metoda vezme podepsanou 4bajtovou celočíselnou hodnotu a zapíše ji do vyrovnávací paměti dat zprávy jako 4bajtové binární číslo začínající na bajtu, na který se odkazuje hodnota DataOffset. Nahrazuje

veškerá data již na těchto pozicích ve vyrovnávací paměti a v případě potřeby rozšiřuje délku vyrovnávací paměti (MQMessage.MessageLength).

Hodnota DataOffset je zvýšena o 4, je-li metoda úspěšná.

Metoda se převede do binární reprezentace zadané vlastností MQMessage.Encoding .

Definováno v: třídě MQMessage

Syntaxe: Volejte *MQMessage*.**WriteLong**(*value &*)

Parametr *hodnota &* Long. Hodnota, která má být zapsána.

Metoda WriteNullTerminatedString

Tato metoda provádí normální WriteString a vycpává zbývající bajty do zadané délky s hodnotou null. Pokud se počet bajtů zapsaných počátečním řetězcem zápisu rovná zadané délce, nezapíší se žádné hodnoty null. Pokud počet bajtů překročí uvedenou délku, bude nastavena chyba (kód příčiny MQRC_WRITE_VALUE_ERROR).

Hodnota DataOffset se zvýší o určenou délku, pokud je metoda úspěšná.

Definováno v: třídě MQMessage

Syntaxe: Volejte *MQMessage*.**WriteNullTerminatedString***(value$, length &)*

Parametry:
hodnotu $String. Hodnota, která má být zapsána.

délka & Long. Délka pole řetězce v bajtech.

Metoda WriteShort

Tato metoda vezme podepsanou 2bajtovou celočíselnou hodnotu a zapíše ji do vyrovnávací paměti dat zprávy jako 2bajtové binární číslo začínající na bajtu, na který se odkazuje DataOffset. Nahrazuje veškerá data již na těchto pozicích ve vyrovnávací paměti a v případě potřeby prodlouží délku vyrovnávací paměti (MQMessage.MessageLength).

Hodnota DataOffset se zvýší o 2, pokud je metoda úspěšná.

Metoda se převede do binární reprezentace zadané vlastností MQMessage.Encoding .

Definováno v: třídě MQMessage

Syntaxe: Volejte *MQMessage*.**WriteShort**(*value%*)

Parametr *hodnota%* celé číslo. Hodnota, která má být zapsána.

Metoda WriteString

Tato metoda vezme řetězec ActiveX a zapíše jej do vyrovnávací paměti dat zprávy začínající na bajtu, na který se odkazuje hodnota DataOffset. Nahrazuje veškerá data již na těchto pozicích ve vyrovnávací paměti a v případě potřeby prodlouží délku vyrovnávací paměti (MQMessage.MessageLength).

Hodnota DataOffset je zvětšena o délku řetězce v bajtech, pokud metoda uspěje.

Metoda převede znaky na kódovou stránku určenou vlastností MQMessage.CharacterSet .

Definováno v: třídě MQMessage

Syntaxe: Volejte *MQMessage*.**WriteString**(*value $*)

Parametr *hodnota $* Řetězec. Hodnota, která má být zapsána.

Metoda WriteUInt2

Tato metoda vezme podepsanou 4bajtovou celočíselnou hodnotu a zapíše ji do vyrovnávací paměti dat zprávy jako 2bajtové binární číslo bez znaménka začínající na bajtu, na který odkazuje hodnota DataOffset.

Nahrazuje veškerá data již na těchto pozicích ve vyrovnávací paměti a v případě potřeby rozšiřuje délku vyrovnávací paměti (MQMessage.MessageLength).

Hodnota DataOffset se zvýší o 2, pokud je metoda úspěšná.

Metoda se převede do binární reprezentace zadané vlastností MQMessage.Encoding . Uvedená hodnota by měla být v rozsahu 0 až 2 * *16-1. Pokud se nejedná o metodu s návratem CompletionCode MQCC_FAILED a ReasonCode MQRC_WRITE_VALUE_ERROR.

Definováno v: třídě MQMessage

Syntaxe: Volat *MQMessage*.**WriteUInt2**(*hodnota &*)

Parametr *hodnota &* Long. Hodnota, která má být zapsána.

WriteUnsignedBytová metoda

Tato metoda vezme podepsanou 2bajtovou celočíselnou hodnotu a zapíše ji do vyrovnávací paměti dat zprávy jako 1-bajtové binární číslo bez znaménka začínající na znaku, na který se odkazuje hodnota DataOffset. Nahrazuje veškerá data již na těchto pozicích ve vyrovnávací paměti a v případě potřeby rozšiřuje délku vyrovnávací paměti (MQMessage.MessageLength).

Hodnota DataOffset je zvýšena o 1, je-li metoda úspěšná.

Uvedená hodnota by měla být v rozsahu 0 až 255. Pokud se nejedná o metodu s návratem CompletionCode MQCC_FAILED a ReasonCode MQRC_WRITE_VALUE_ERROR.

Definováno v:
Třída MQMessage

Syntaxe: Volejte *MQMessage*.**WriteUnsignedByte**(*value%*)

Parametr *hodnota%* celé číslo. Hodnota, která má být zapsána.

Metoda WriteUTF

Tato metoda vezme řetězec ActiveX a zapíše jej do vyrovnávací paměti dat zprávy na aktuální pozici ve formátu UTF. Zapsaných dat se skládá z 2bajtové délky následované znakovými daty. Hodnota DataOffset je zvětšena o délku řetězce, pokud metoda uspěje.

Definováno v:
Třída MQMessage

Syntaxe: Volat *MQMessage*.**WriteUTF**(value$)

Parametr:
hodnota $String. Hodnota, která má být zapsána.

Třída voleb MQPutMessage

Tato třída zapouzdřuje různé volby, které řídí akci vložení zprávy do fronty produktu WebSphere MQ .

Obsažení

Třída voleb MQPutMessageje obsažena ve třídě MQSession.

VYTVOŘENÍ

Volba **Nový** vytvoří nový objekt voleb MQPutMessagea nastaví všechny jeho vlastnosti na počáteční hodnoty.

Případně můžete použít metodu AccessPutMessageOptions třídy MQSession.

Syntaxe

Prom *pmo* **Jako Nový MQPutMessage** nebo

Nastavit *pmo* = **Nový VolbyMQPutMessage**

Vlastnosti

- "Vlastnost CompletionCode" na stránce 1058.
- "Vlastnost Options" na stránce 1058.
- "Vlastnost ReasonCode" na stránce 1058.
- "Vlastnost ReasonName" na stránce 1058.
- "Vlastnost RecordFields" na stránce 1059.
- "Vlastnost ResolvedQueueManagerName" na stránce 1059.
- "Vlastnost názvu ResolvedQueue" na stránce 1059.

Metody

- "Metoda ClearErrorCodes" na stránce 1059.

Vlastnost CompletionCode

Pouze pro čtení. Vrací kód dokončení nastavený podle poslední metody nebo přístupu k vlastnosti, který byl vydán proti objektu.

Definováno v: třídě voleb MQPutMessage

Datový typ: Long

Hodnoty:

- MQCC_OK
- VAROVÁNÍ MQCC_WARNING
- SELHÁNÍ MQCC_FAILED

Syntaxe: Chcete-li získat: *completioncode & = PutOpts*.**CompletionCode**

Vlastnost Options

Čtení a zápis. Pole Volby MQPMO. Počáteční hodnota tohoto pole je MQPMO_NONE. Další informace naleznete v tématu Volby MQPMO.

Definováno v: třídě voleb MQPutMessage.

Datový typ: Long

Syntaxe: Chcete-li získat: *volby & = PutOpts*.**Volby**

Chcete-li nastavit: *PutOpts*.**Volby** = *volby &*

Volby MQPMO_PASS_IDENTITY_CONTEXT a MQPMO_PASS_ALL_CONTEXT nejsou podporovány.

Vlastnost ReasonCode

Pouze pro čtení. Vrátí kód příčiny nastavený posledním voláním metody nebo vlastností, které byly vydány pro daný objekt.

Definováno v: třídě voleb MQPutMessage

Datový typ: Long

Hodnoty:

- Viz termín kód příčiny rozhraní API.

Syntaxe: Chcete-li získat následující informace: *reasoncode & = PutOpts*.**ReasonCode**

Vlastnost ReasonName

Pouze pro čtení. Vrátí symbolický název posledního kódu příčiny. Příklad: "MQRC_QMGR_NOT_AVAILABLE".

Definováno v: třídě voleb MQPutMessage

Datový typ: Řetězec

Hodnoty:

- Viz termín kód příčiny rozhraní API.

Syntaxe: Chcete-li získat: *reasonname $= PutOpts*.**ReasonName**

Vlastnost RecordFields

Čtení a zápis. Příznaky označující, která pole mají být při vložení zprávy do distribučního seznamu přizpůsobena pro jednotlivé fronty. Počáteční hodnota je nula.

Tato vlastnost odpovídá příznaky PutMsgRecFields ve struktuře MQPMO rozhraní MQI MQI. V rozhraní MQI tyto příznaky řídí, která pole (ve struktuře MQPMR) jsou přítomna a používána operací MQPUT. V objektu Volby MQPutMessagejsou tato pole vždy přítomná a parametry proto ovlivňují pouze ta pole, která jsou použita vložením. Další podrobnosti naleznete v příručce *WebSphere MQ Application Programming Reference* .

Definováno v:
Třída voleb MQPutMessage

Datový typ:
Dlouhý

Syntaxe: Chcete-li získat: *recordfields & = PutOpts*.**RecordFields**

Chcete-li nastavit: *PutOpts*.**RecordFields** = *recordfields &*

Vlastnost ResolvedQueueManagerName

Pouze pro čtení. Pole názvu MQPMO ResolvedQMgr. Podrobnosti najdete v tématu ResolvedQMgrName (MQCHAR48) . Počáteční hodnota je prázdná.

Definováno v: třídě voleb MQPutMessage

Typ dat: Řetězec 48 znaků

Syntaxe: Chcete-li získat: *qmgr $= PutOpts*.**ResolvedQueueManagerName**

Vlastnost názvu ResolvedQueue

Pouze pro čtení. Pole MQPMO ResolvedQName . Podrobnosti viz ResolvedQName (MQCHAR48) . Počáteční hodnota je prázdná.

Definováno v: třídě voleb MQPutMessage

Typ dat: Řetězec 48 znaků

Syntaxe: Chcete-li získat: *qname $= PutOpts*.**ResolvedQueueName**

Metoda ClearErrorCodes

Resetuje parametr CompletionCode na MQCC_OK a ReasonCode na hodnotu MQRC_NONE jak pro třídu voleb MQPutMessage, tak pro třídu MQSession.

Definováno v: třídě voleb MQPutMessage

Syntaxe: Call *PutOpts*.**ClearErrorCodes ()**

Třída voleb MQGetMessage

Tato třída zapouzdřuje různé volby, které řídí akci získání zprávy z fronty produktu WebSphere MQ .

Obsažení

Třída voleb MQGetMessageje obsažena ve třídě MQSession.

VYTVOŘENÍ

Volba **Nový** vytvoří nový objekt voleb MQGetMessagea nastaví všechny její vlastnosti na počáteční hodnoty.

Případně můžete použít metodu AccessGetMessageOptions třídy MQSession.

Vlastnosti

Metody

Syntaxe

Dim *gmo* **Jako nové volby MQGetMessage** nebo

Nastavit *gmo* = **Nové volby MQGetMessage**

Vlastnost CompletionCode

Pouze pro čtení. Vrací kód dokončení nastavený podle poslední metody nebo přístupu k vlastnosti, který byl vydán proti objektu.

Definováno v: třídě voleb MQGetMessage.

Datový typ: Long

Hodnoty:

- MQCC_OK
- VAROVÁNÍ MQCC_WARNING
- SELHÁNÍ MQCC_FAILED

Syntaxe: Chcete-li získat: *completioncode & = GetOpts***.CompletionCode**

Vlastnost MatchOptions

Čtení a zápis. Volby, které řídí kritéria výběru použitá pro MQGET. Počáteční hodnota je MQMO_MATCH_MSG_ID + MQMO_MATCH_CORREL_ID.

Definováno v:
Třída voleb MQGetMessage

Datový typ:
Dlouhý

Hodnoty:
Viz MatchOptions (MQLONG).

Syntaxe: Chcete-li získat: *matchoptions & = GetOpts*.**MatchOptions**

Pro nastavení: *GetOpts*.**MatchOptions** =*matchoptions &*

Vlastnost Options

Čtení a zápis. Pole Volby MQGMO. Podrobnosti viz Volby . Počáteční hodnota je MQGMO_NO_WAIT.

Definováno v: třídě voleb MQGetMessage.

Datový typ: Long

Syntaxe: Chcete-li získat: *volby & = GetOpts***.Volby** Chcete-li nastavit: *GetOpts***.Volby** = *volby &*

Vlastnost ReasonCode

Pouze pro čtení. Vrátí kód příčiny nastavený posledním voláním metody nebo vlastností, které byly vydány pro daný objekt.

Definováno v: třídě voleb MQGetMessage

Datový typ: Long

Hodnoty:

- Viz termín kód příčiny rozhraní API.

Syntaxe: Chcete-li získat následující informace: *reasoncode & = GetOpts***.ReasonCode**

Vlastnost ReasonName

Pouze pro čtení. Vrátí symbolický název posledního kódu příčiny. Příklad: "MQRC_QMGR_NOT_AVAILABLE". **Definováno v:** třídě voleb MQGetMessage

Datový typ: Řetězec

Hodnoty:

- Viz termín kód příčiny rozhraní API.

Syntaxe: Chcete-li získat: *reasonname $= MQGetMessageOptions***.ReasonName**

Vlastnost názvu ResolvedQueue

Pouze pro čtení. Pole MQGMO ResolvedQName . Podrobnosti viz ResolvedQName (MQCHAR48) . Počáteční hodnota je prázdná.

Definováno v: třídě voleb MQGetMessage

Typ dat: Řetězec 48 znaků

Syntaxe: Chcete-li získat: *qname $= GetOpts***.ResolvedQueueName**

Vlastnost WaitInterval

Čtení/zápis. Pole MQGMO WaitInterval . Maximální doba v milisekundách, po kterou komponenta Get čeká na vhodnou zprávu-pokud byla akce čekání požadována vlastností Options. Toto pole má počáteční hodnotu 0. Podrobné informace o volbách MQGMO naleznete v tématu MQGMO.

Definováno v: třídě voleb MQGetMessage

Datový typ: Long

Syntaxe: Chcete-li získat: *wait & = GetOpts***.WaitInterval**

Pro nastavení: *GetOpts***.WaitInterval** = wait &

Metoda ClearErrorCodes

Resetuje parametr CompletionCode na MQCC_OK a ReasonCode na MQRC_NONE jak pro třídu voleb MQGetMessage, tak pro třídu MQSession.

Definováno v: třídě voleb MQGetMessage

Syntaxe: Call *GetOpts*.**ClearErrorCodes ()**

Třída MQDistributionList

Tato třída zapouzdřuje kolekci front-lokálních, vzdálených nebo aliasů pro výstup.

VYTVOŘENÍ

new vytvoří nový objekt MQDistributionList .

Případně můžete použít metodu AddDistributionList třídy MQQueueManager .

Vlastnosti

- "Vlastnost ID AlternateUser" na stránce 1062
- "Vlastnost CloseOptions" na stránce 1062
- "Vlastnost CompletionCode" na stránce 1063
- "Vlastnost ConnectionReference" na stránce 1063
- "Vlastnost FirstDistributionListItem" na stránce 1063
- "Vlastnost IsOpen" na stránce 1063
- "Vlastnost OpenOptions" na stránce 1064
- "Vlastnost ReasonCode" na stránce 1064
- "Vlastnost ReasonName" na stránce 1064

Metoda

- "Metoda AddDistributionListItem" na stránce 1064
- "Metoda ClearErrorCodes" na stránce 1065
- "Metoda Close" na stránce 1065
- "Otevřít metodu" na stránce 1065
- "metoda PUT" na stránce 1065

Syntaxe

Dim *seznam_distr.***As novými MQDistributionList** nebo **Set** *distlist* = **New MQDistributionList**

Vlastnost ID AlternateUser

Čtení a zápis. Alternativní ID uživatele použité pro ověření přístupu k seznamu front, když jsou otevřeny.

Definováno v:
Třída MQDistributionList

Datový typ:
Řetězec 12 znaků

Syntaxe: Chcete-li získat: *altuser $= MQDistributionList*.**AlternateUserId**

Chcete-li nastavit: *MQDistributionList*.**AlternateUserId** = *altuser $*

Vlastnost CloseOptions

Čtení a zápis. Volby používané k řízení toho, co se stane, když se distribuční seznam uzavře. Počáteční hodnota je MQCO_NONE.

Definováno v:
Třída MQDistributionList

Datový typ:
Dlouhý

Hodnoty:

- MQCO_NONE
- MQCO_DELETE
- MQCO_DELETE_PURGE

Syntaxe: Chcete-li získat následující informace: *closeopt & = MQDistributionList*.**CloseOptions**

Chcete-li nastavit: *MQDistributionList*.**CloseOptions** = *closeopt &*

Vlastnost CompletionCode

Pouze pro čtení. Kód dokončení nastavený pomocí poslední metody nebo přístupu k vlastnosti vydaného pro objekt.

Definováno v:
Třída MQDistributionList

Datový typ:
Dlouhý

Hodnoty:

- MQCC_OK
- VAROVÁNÍ MQCC_WARNING
- SELHÁNÍ MQCC_FAILED

Syntaxe: Chcete-li získat: *completioncode & = MQDistributionList*.**CompletionCode**

Vlastnost ConnectionReference

Čtení a zápis. Správce front, do kterého patří distribuční seznam.

Definováno v:
Třída MQDistributionList

Datový typ:
MQQueueManager

Syntaxe: Chcete-li získat: *set queuemanager = MQDistributionList*.**ConnectionReference**

Nastavení: *set MQDistributionList*. **ConnectionReference** = *queuemanager*

Vlastnost FirstDistributionListItem

Pouze pro čtení. První objekt položky rozdělovníku přidružený k rozdělovníku.

Definováno v:
Třída MQDistributionList

Datový typ:
Položka MQDistributionList

Hodnoty:

Syntaxe: Chcete-li získat: *set distributionlistitem = MQDistributionList*.**FirstDistributionListItem**

Vlastnost IsOpen

Pouze pro čtení.

Definováno v:
Třída MQDistributionList

Datový typ:
Logická hodnota

Hodnoty:

- TRUE (-1)
- FALSE (0)

Syntaxe: Chcete-li získat: *IsOpen = MQDistributionList*.**IsOpen**

Vlastnost OpenOptions

Čtení a zápis. Volby, které se mají použít při otevření distribučního seznamu.

Definováno v:
Třída MQDistributionList

Datový typ:
Dlouhý

Hodnoty:
Viz Volby MQPMO.

Syntaxe: Chcete-li získat následující informace: *openopt & = MQDistributionList*.**OpenOptions**

Chcete-li nastavit: *MQDistributionList.***OpenOptions** = openopt &

Vlastnost ReasonCode

Pouze pro čtení. Kód příčiny nastavený pomocí poslední metody nebo přístupu k vlastnosti vydaného pro objekt.

Definováno v:
Třída MQDistributionList

Datový typ:
Dlouhý

Hodnoty:
Viz termín kód příčiny rozhraní API.

Syntaxe: Chcete-li získat následující informace: *reasoncode & = MQDistributionList*.**ReasonCode**

Vlastnost ReasonName

Pouze pro čtení. Symbolický název pro ReasonCode. Například "MQRC_QMGR_NOT_AVAILABLE".

Definováno v:
Třída MQDistributionList

Datový typ:
Řetězec

Hodnoty:
Viz termín kód příčiny rozhraní API.

Syntaxe: Chcete-li získat: *reasonname $= MQDistributionList*.**ReasonName**

Metoda AddDistributionListItem

Vytvoří nový objekt položky MQDistributionLista přidruží jej k objektu seznamu distribuce. Parametr názvu fronty je povinný.

Vlastnost DistributionList položky rozdělovníku je nastavena na vlastnící distribuční seznam a vlastnost FirstDistributionListItem distribučního seznamu je nastavena tak, aby odkazovaly na tuto novou položku distribučního seznamu.

Pro novou položku distribučního seznamu je vlastnost PreviousDistributionListItem nastavena na nic a vlastnost NextDistributionListItem je nastavena tak, aby odkazovalo na položku seznamu distribuce, která byla dříve první, nebo nic, pokud předtím nebyla žádná (tj. nová položka je vložena před již existující).

To vrátí chybu, pokud je distribuční seznam otevřený.

Definováno v:
Třída MQDistributionList

Syntaxe: set distributionlistitem = *MQDistributionList*.**AddDistributionListItem** (QName$, QMgrName$)

Parametry:

Řetězec *QName$* . Název fronty produktu WebSphere MQ .

Řetězec *QMgrName$* String. Název správce front produktu WebSphere MQ .

Metoda ClearErrorCodes

Resetuje parametr CompletionCode na MQCC_OK a ReasonCode na MQRC_NONE jak pro třídu MQDistributionList , tak pro třídu MQSession.

Definováno v:
Třída MQDistributionList

Syntaxe: Volejte *MQDistributionList*.**ClearErrorCodes**()

Metoda Close

Zavře distribuční seznam s použitím aktuální hodnoty voleb Zavřít.

Definováno v:
Třída MQDistributionList

Syntaxe: Volejte *MQDistributionList*.**Close**()

Otevřít metodu

Otevře každou z front uvedených ve vlastnostech QueueName a (kde je to vhodné) QueueManagerNázev položek distribučního seznamu přidruženého k aktuálnímu objektu s použitím aktuální hodnoty ID AlternateUser.

Definováno v:
Třída MQDistributionList

Syntaxe: Volejte *MQDistributionList*.**Open**()

metoda PUT

Umístí zprávu na každou z front uvedených v položkách rozdělovníku přidružených k rozdělovníku.

Definováno v:

Třída MQDistributionList

Syntaxe

Volejte MQDistributionList.**Put**(Zpráva, volby PutMsg&)

Parametry

Objekt *Message* MQMessage představující zprávu, která má být vložena.

Objekt *PutMsgOptions* MQPutMessageOptions obsahující volby pro řízení operace put. Není-li tento parametr zadán, použijí se výchozí volby PutMessage.

Tato metoda přijímá objekt MQMessage jako parametr. Jako výsledek této metody lze změnit následující vlastnosti položky seznamu distribucí:

- CompletionCode
- ReasonCode
- ReasonName
- MessageId
- MessageIdHex
- CorrelationId
- CorrelationIdHexadecimální číslo
- GroupId
- GroupIdHexadecimální
- Zpětná vazba
- AccountingToken
- AccountingTokenHex

Třída položek MQDistributionList

Tato třída zapouzdřuje struktury MQOR, MQRR a MQPMR a sdružuje je s vlastnícím distribučním seznamem.

VYTVOŘENÍ

Použití metody AddDistributionListItem třídy MQDistributionList

Vlastnosti

Metody

- “Vlastnost AccountingToken” na stránce 1067.
- “AccountingTokenHexadecimální vlastnost” na stránce 1067.
- “Vlastnost CompletionCode” na stránce 1068.
- “Vlastnost CorrelationId” na stránce 1068.
- “CorrelationIdHexadecimální vlastnost” na stránce 1068.
- “Vlastnost DistributionList” na stránce 1068.
- “Vlastnost Feedback” na stránce 1069.
- “Vlastnost GroupId” na stránce 1069.
- “GroupIdHexadecimální vlastnost” na stránce 1069.
- “Vlastnost MessageId” na stránce 1069.
- “MessageIdHexadecimální vlastnost” na stránce 1069.
- “Vlastnost NextDistributionListItem” na stránce 1070.
- “Vlastnost PreviousDistributionListItem” na stránce 1070.
- “Vlastnost názvu QueueManager” na stránce 1070.
- “Vlastnost QueueName” na stránce 1070.
- “Vlastnost ReasonCode” na stránce 1070.
- “Vlastnost ReasonName” na stránce 1071.
- “Metoda ClearErrorCodes” na stránce 1071.

Vlastnosti:

- Vlastnost AccountingToken
- AccountingTokenHexadecimální vlastnost
- Vlastnost CompletionCode
- Vlastnost CorrelationId
- CorrelationIdHexadecimální vlastnost
- Vlastnost DistributionList
- Vlastnost Feedback
- Vlastnost GroupId
- GroupIdHexadecimální vlastnost
- Vlastnost MessageId
- MessageIdHexadecimální vlastnost
- Vlastnost NextDistributionListItem
- Vlastnost PreviousDistributionListItem
- Vlastnost názvu QueueManager
- Vlastnost QueueName
- Vlastnost ReasonCode
- Vlastnost ReasonName

Metody:

- Metoda ClearErrorCodes

Vytvoření:

Použití metody AddDistributionListItem třídy MQDistributionList

Vlastnost AccountingToken

Čtení a zápis. Hodnota AccountingToken bude zahrnuta do zprávy MQPMR zprávy při vložení do fronty. Jeho počáteční hodnota je všechny hodnoty null.

Definováno v:
Třída položek MQDistributionList

Datový typ:
Řetězec 32 znaků

Syntaxe: Chcete-li získat: *accountingtoken $= MQDistributionListItem*.**AccountingToken**

Chcete-li nastavit: *MQDistributionListPoložka*.**AccountingToken** = *accountingtoken $*

AccountingTokenHexadecimální vlastnost

Čtení a zápis. Hodnota AccountingToken bude zahrnuta do zprávy MQPMR zprávy při vložení do fronty.

Každé dva znaky řetězce reprezentují hexadecimální ekvivalent jednoho znaku ASCII. Například dvojice znaků "6" a "1" představují jediný znak "A", dvojice znaků "6" a "2" představují jediný znak "B" a tak dále.

Je třeba zadat 64 platných hexadecimálních znaků.

Jeho počáteční hodnota je "0 ... 0".

Definováno v:
Třída položek MQDistributionList

Datový typ:
Řetězec obsahující 64 hexadecimálních znaků, který požaduje 32 znaků ASCII.

Syntaxe: Chcete-li získat: *accountingtokenh $= MQDistributionListItem*.**AccountingTokenHex**

Nastavení: *MQDistributionListItem*.**AccountingTokenHex** = *accountingtokenh $*

Vlastnost CompletionCode

Pouze pro čtení. Kód dokončení nastavený posledním otevřeným vydáním nebo požadavkem na vložení vydaného na vlastnící objekt distribučního seznamu.

Definováno v:
Třída položek MQDistributionList

Datový typ:
Dlouhý

Hodnoty:

- MQCC_OK
- VAROVÁNÍ MQCC_WARNING
- SELHÁNÍ MQCC_FAILED

Syntaxe: Chcete-li získat: *completioncode $= MQDistributionListItem*.**CompletionCode**

Vlastnost CorrelationId

Čtení a zápis. CorrelId , které má být zahrnuto do MQPMR zprávy při vložení do fronty. Jeho počáteční hodnota je všechny hodnoty null.

Definováno v:
Třída položek MQDistributionList

Datový typ:
Řetězec o délce 24 znaků

Syntaxe: Chcete-li získat: *correlid $= MQDistributionListItem*.**CorrelationId**

Chcete-li nastavit: *MQDistributionListItem*.**CorrelationId** = *correlid $*

CorrelationIdHexadecimální vlastnost

Čtení a zápis. CorrelId , které má být zahrnuto do MQPMR zprávy při vložení do fronty.

Každé dva znaky řetězce reprezentují hexadecimální ekvivalent jednoho znaku ASCII. Například dvojice znaků "6" a "1" představují jediný znak "A", dvojice znaků "6" a "2" představují jediný znak "B" a tak dále.

Je třeba zadat 48 platných hexadecimálních znaků.

Jeho počáteční hodnota je "0 .. 0".

Definováno v:
Třída položek MQDistributionList

Datový typ:
Řetězec 48 hexadecimálních znaků představujících 24 znaků ASCII.

Syntaxe: Chcete-li získat: *correlidh $= MQDistributionListItem*.**CorrelationIdHex**

Chcete-li nastavit: *MQDistributionListItem*.**CorrelationIdHex** = *correldh $*

Vlastnost DistributionList

Pouze pro čtení. Distribuční seznam, se kterým je tato položka rozdělovníku přidružena.

Definováno v:
Třída položek MQDistributionList

Datový typ:
MQDistributionList

Syntaxe: Chcete-li získat: *set distributionlist = MQDistributionListItem*.**DistributionList**

Vlastnost Feedback

Čtení a zápis. Hodnota zpětné vazby, která má být zahrnuta do zprávy MQPMR zprávy při vložení do fronty.

Definováno v:
Třída položek MQDistributionList

Datový typ:
Dlouhý

Hodnoty:
Viz Zpětná vazba (MQLONG).

Syntaxe: Chcete-li získat: *feedback & = MQDistributionListItem*.**Feedback**

Chcete-li nastavit: *MQDistributionListItem*.**Feedback** = *feedback &*

Vlastnost GroupId

Čtení a zápis. Hodnota GroupId , která má být zahrnuta do zprávy MQPMR zprávy při vložení do fronty. Jeho počáteční hodnota je všechny hodnoty null.

Definováno v:
Třída položek MQDistributionList

Datový typ:
Řetězec o délce 24 znaků

Syntaxe: Chcete-li získat: *groupid $= MQDistributionListItem*.**GroupId**

Chcete-li nastavit: *MQDistributionListItem*.**GroupId** = *groupid $*

GroupIdHexadecimální vlastnost

Čtení a zápis. Hodnota GroupId , která má být zahrnuta do zprávy MQPMR zprávy při vložení do fronty.

Každé dva znaky řetězce reprezentují hexadecimální ekvivalent jednoho znaku ASCII. Například dvojice znaků "6" a "1" představují jediný znak "A", dvojice znaků "6" a "2" představují jediný znak "B" a tak dále.

Je třeba zadat 48 platných hexadecimálních znaků.

Jeho počáteční hodnota je "0 .. 0".

Definováno v:
Třída položek MQDistributionList

Datový typ:
Řetězec 48 hexadecimálních znaků reqpresenting 24 znaků ASCII.

Syntaxe: Chcete-li získat: *groupidh $= MQDistributionListItem*.**GroupIdHex**

Chcete-li nastavit: *MQDistributionListItem*.**GroupIdHex** = *groupidh $*

Vlastnost MessageId

Čtení a zápis. Hodnota MessageId , která má být zahrnuta do zprávy MQPMR zprávy při vložení do fronty. Jeho počáteční hodnota je všechny hodnoty null.

Definováno v:
Třída položek MQDistributionList

Datový typ:
Řetězec o délce 24 znaků

Syntaxe: Chcete-li získat: *messageid $= MQDistributionListItem*.**MessageId**

Nastavení: *MQDistributionListItem*.**MessageId** = *messageid $*

MessageIdHexadecimální vlastnost

Čtení a zápis. Hodnota MessageId , která má být zahrnuta do zprávy MQPMR zprávy při vložení do fronty.

Každé dva znaky řetězce reprezentují hexadecimální ekvivalent jednoho znaku ASCII. Například dvojice znaků "6" a "1" představují jediný znak "A", dvojice znaků "6" a "2" představují jediný znak "B" a tak dále.

Je třeba zadat 48 platných hexadecimálních znaků.

Jeho počáteční hodnota je "0 .. 0".

Definováno v:
Třída položek MQDistributionList

Datový typ:
Řetězec 48 hexadecimálních znaků představujících 24 znaků ASCII.

Syntaxe: Chcete-li získat: *messagesh $= MQDistributionListItem*.**MessageIdHex**

Nastavení: *MQDistributionListItem*.**MessageIdHex** = *messageidh $* .

Vlastnost NextDistributionListItem

Pouze pro čtení. Další objekt položky rozdělovníku přidružený ke stejnému rozdělovníku.

Definováno v:
Třída položek MQDistributionList

Datový typ:
Položka MQDistributionList

Syntaxe: Chcete-li získat: *set distributionlistitem = MQDistributionListItem*.**NextDistributionListItem**

Vlastnost PreviousDistributionListItem

Pouze pro čtení. Předchozí objekt položky seznamu distribuce přidružený ke stejnému rozdělovníku.

Definováno v:
Třída položek MQDistributionList

Datový typ:
Položka MQDistributionList

Syntaxe: Chcete-li získat: *set distributionlistitem = MQDistributionListItem*.**PreviousDistributionListItem**

Vlastnost názvu QueueManager

Čtení a zápis. Název správce front produktu WebSphere MQ .

Definováno v:
Třída položek MQDistributionList

Datový typ:
Řetězec 48 znaků.

Syntaxe: Chcete-li získat: *qmname $= MQDistributionListItem*.**QueueManagerName**

Chcete-li nastavit: *MQDistributionListItem*.**QueueManagerName** = *qmname $*

Vlastnost QueueName

Čtení a zápis. Název fronty produktu WebSphere MQ .

Definováno v:
Třída položek MQDistributionList

Datový typ:
Řetězec 48 znaků.

Syntaxe: Chcete-li získat: *qname $= MQDistributionListItem*.**QueueName**

Chcete-li nastavit: *MQDistributionListItem*.**QueueName** = *qname $*

Vlastnost ReasonCode

Pouze pro čtení. Kód dokončení nastavený pomocí posledního otevření nebo vložení vydaného na vlastnící objekt distribučního seznamu.

Definováno v:
Třída položek MQDistributionList

Datový typ:
Dlouhý

Hodnoty:
Viz termín kód příčiny rozhraní API.

- MQCC_OK
- VAROVÁNÍ MQCC_WARNING
- SELHÁNÍ MQCC_FAILED

Syntaxe: Chcete-li získat: *reasoncode & = MQDistributionListItem*.**ReasonCode**

Vlastnost ReasonName

Pouze pro čtení. Symbolický název pro ReasonCode. Například "MQRC_QMGR_NOT_AVAILABLE".

Definováno v:
Třída položek MQDistributionList

Datový typ:
Řetězec

Hodnoty:
Viz termín kód příčiny rozhraní API.

Syntaxe: Chcete-li získat: *reasonname $= MQDistributionListItem*.**ReasonName**

Metoda ClearErrorCodes

Resetuje parametr CompletionCode na MQCC_OK a ReasonCode na MQRC_NONE jak pro třídu položek MQDistributionList, tak pro třídu MQSession.

Definováno v:
Třída položek MQDistributionList

Syntaxe: Volejte položku *MQDistributionListItem*.**ClearErrorCodes**

Odstraňování problémů

Informace o poskytnutém trasovacím zařízení, společná nástrahy a pomoc s tím, jak se jim vyhnout.

Tento oddíl vysvětluje poskytovanou službu trasování a podrobnosti o úskalí, které jim pomohou vyhnout se jim:

- “Použití trasování” na stránce 1071
- “Pokud se nezdaří skript WebSphere MQ Automation Classes for ActiveX” na stránce 1072
- “Kódy příčin” na stránce 1073
- “Nástroj úrovně kódu” na stránce 1075

Použití trasování

MQAX zahrnuje trasovací prostředek, který pomáhá organizaci služeb identifikovat to, co se děje, když se vyskytl problém. Zobrazuje cesty, které se provedou při spuštění skriptu MQAX. Pokud nemáte problém, spusťte s vypnutím trasování, abyste se vyhnuli zbytečnému využití systémových prostředků.

Existují tři proměnné prostředí, které jste nastavili pro řízení trasování:

- TRASOVÁNÍ OMQ_
- CESTOVACÍ_CESTA VYNECHÁNÍ

- OMQ_TRACE_LEVEL

Všimněte si, že zadání hodnoty *any* pro volbu OMQ_TRACE přepíná trasovací prostředek na zapnuto. I když nastavíte volbu ODMQ_TRACE na hodnotu OFF, trasování je stále aktivní.

Chcete-li vypnout trasování, neuvádějte hodnotu pro volbu OMQ_TRACE.

1. Klepněte na tlačítko **Spustit**
2. Klepněte na **Ovládací panely**
3. Dvakrát klepněte na **Systém**
4. Klepněte na **Rozšířené** .
5. Klepněte na volbu **Prostředí**
6. V sekci s názvem "Uživatelské proměnné pro (jméno uživatele)" klepněte na volbu **Nový** .
7. Zadejte název proměnné a platnou hodnotu do příslušných polí a klepněte na tlačítko **OK** .
8. Klepnutím na tlačítko **OK** zavřete okno Proměnné prostředí.
9. Klepnutím na tlačítko **OK** zavřete okno Systémové vlastnosti.
10. Zavřít okno Ovládací panely

Při rozhodování o tom, kam chcete trasovací soubory zapsat, ujistěte se, že máte dostatečné oprávnění k zápisu na disk, nejen pro čtení, ale také pro čtení.

Se zapnutým trasováním zpomaluje běh MQAX, ale neovlivňuje výkon prostředí ActiveX nebo prostředí WebSphere MQ . Pokud již trasovací soubor nepotřebujete, můžete jej odstranit.

Chcete-li změnit stav proměnné ODMQ_TRACE, musíte zastavit aplikaci MQAX.

Název a adresář trasovacího souboru

Název trasovacího souboru má tvar OMQnnnnn.trc, kde nnnnn je ID procesu ActiveX , který běží v daném okamžiku.

Tabulka 157. Příkazy a jejich efekty

Příkaz	Efekt
SET OMQ_TRACE_PATH = jednotka: \adresář	Nastaví trasovací adresář, do kterého bude zapsán trasovací soubor.
NASTAVIT CESTU OQ_TRACE_PATH =	Odstraní proměnnou prostředí ODMQ_PATH tak, aby byl použit aktuální pracovní adresář (je-li spuštěn ActiveX).
ECHO %OMQ_TRACE_PATH%	Zobrazí aktuální nastavení trasovacího adresáře v systému Windows.
SET OMQ_TRACE = xxxxxxxx	Tím nastavíte trasování ON. Vypíšete trasování tak, že vložíte jeden nebo více znaků za znak '='. Například: SET OMQ_TRACE=yes SET OMQ_TRACE = no. V obou těchto příkladech bude trasování nastaveno na hodnotu ON. Toto je platné pouze pro jedno okno/relaci okna.
NASTAVIT FUNKCI OQ_TRACE=	Vypne trasování
ECHO %OMQ_TRACE%	Zobrazí obsah proměnné prostředí v systému Windows.
SET	Zobrazí obsah všech proměnných prostředí v systému Windows.
NASTAVIT PARAMETR OMQ_TRACE_LEVEL = 9	Nastavuje úroveň trasování na hodnotu 9. Hodnoty větší než 9 nevytvářejí žádné další informace v trasovacím souboru.

Pokud se nezdaří skript WebSphere MQ Automation Classes for ActiveX

Pokud váš skript WebSphere MQ Automation Classes for ActiveX selže, je zde mnoho zdrojů informací.

Hlášení projevů prvního selhání

Nezávisle na mechanizmu trasování, v případě neočekávaných a interních chyb, může být vytvořena zpráva o selhání prvního selhání.

Tato sestava se nachází v souboru s názvem OMQnnnnn.fdc, kde nnnnn je počet procesů ActiveX , které jsou spuštěny v daném okamžiku. Tento soubor najdete v pracovním adresáři, ze kterého jste spustili ActiveX nebo v cestě uvedené v proměnné prostředí OMQ_PATH.

Další zdroje informací

Produkt WebSphere MQ poskytuje různé protokoly chyb a trasovací informace v závislosti na použité platformě. Prohlédněte si protokol událostí aplikace Windows NT.

Kódy příčin

Kromě těch, které jsou zdokumentovány pro rozhraní WebSphere MQ MQI, se mohou vyskytnout následující kódy příčiny. Informace o dalších kódech najdete v protokolu událostí aplikace WebSphere MQ .

Tabulka 158. Kódy příčiny a o tom, co znamenají

Kód příčiny	Vysvětlení
MQRC_LIBRARY_LOAD_ERROR (6000)	Jednu nebo více knihoven produktu WebSphere MQ nebylo možné načíst. Zkontrolujte, že všechny knihovny produktu WebSphere MQ jsou ve správné vyhledávací cestě v systému, který používáte. Ujistěte se například, že adresáře obsahující knihovny produktu WebSphere MQ jsou v proměnné PATH.
CHYBA MQRC_CLASS_LIBRARY_ERROR (6001)	Jeden z volání třídy třídy WebSphere MQ vrátil neočekávanou hodnotu ReasonCode/CompletionCode. Podrobnosti naleznete v sestavě projevů prvního selhání. Vezměte na vědomí poslední použitou metodu/vlastnost a třídu a informujte IBM o problému podporu.
MQRC_STRING_LENGTH_TOO_BIG (6002)	Byl proveden pokus o zápis řetězce formátu UTF s délkou větší než 65 535 bajtů do vyrovnávací paměti zpráv.
CHYBA MQRC_WRITE_VALUE_ERROR (6003)	Je použita hodnota, která je mimo rozsah; například msg.WriteByte (240).
MQRC_PACKED_DECIMAL_ERROR (6004)	Byl proveden pokus o čtení zabaleného dekadického čísla z vyrovnávací paměti zpráv, ale data na datovém ukazateli nejsou v platném zhuštěném formátu dat.
MQRC_FLOAT_CONVERSION_ERROR (6005)	Byl proveden pokus o čtení jednoho nebo dvojnásobného čísla s pohyblivou řádovou čárkou z vyrovnávací paměti zpráv, ale data na ukazateli dat nejsou v náležitém formátu s plovoucí řádovou čárkou.
MQRC_REOPEN_EXCL_INPUT_ERROR (6100)	Otevřený objekt nemá správný **OpenOptions** a vyžaduje jednu nebo více dalších voleb. Je požadováno implicitní znovuotevření, ale uzavření bylo zabráněno. Nastavte volbu **OpenOptions**explicitně tak, aby pokryla všechny možnosti tak, aby implicitní opětovné otevření nebylo požadováno. Uzavírání bylo zabráněno, protože fronta je otevřená pro výlučný vstup a uzavření by prezentovalo okno příležitostí pro ostatní potenciálně získat přístup do fronty.
MQRC_REOPEN_INQUIRE_ERROR (6101)	Otevřený objekt nemá správné OpenOptions a vyžaduje jednu nebo více dalších voleb. Je požadováno implicitní znovuotevření, ale uzavření bylo zabráněno. Nastavte parametr OpenOptions explicitně, aby zahrnoval MQOO_INQUIRE. Uzavírání bylo zabráněno, protože jedna nebo více charakteristik objektu musí být zkontrolována dynamicky před uzavřením, a OpenOptions již neobsahuje MQOO_INQUIRE.

Tabulka 158. Kódy příčiny a o tom, co znamenají (pokračování)

Kód příčiny	Vysvětlení
MQRC_REOPEN_SAVED_CONTEXT_ERR (6102)	Otevřený objekt nemá správné OpenOptions a vyžaduje jednu nebo více dalších voleb. Je požadováno implicitní znovuotevření, ale uzavření bylo zabráněno. Nastavte volbu OpenOptions explicitně tak, aby pokryla všechny možnosti tak, aby implicitní opětovné otevření nebylo požadováno. Uzavírání bylo zabráněno, protože fronta je otevřená s MQOO_SAVE_ALL_CONTEXT a destruktivní operace Get byla již provedena dříve. To způsobilo, že uchovávané informace o stavu byly přidruženy k otevřené frontě a tyto informace by byly zničeny uzavřením.
MQRC_REOPEN_TEMPORARY_Q_ERROR (6103)	Otevřený objekt nemá správný **OpenOptions** a vyžaduje jednu nebo více dalších voleb. Je požadováno implicitní znovuotevření, ale zavření bylo zabráněno. Nastavte volbu **OpenOptions**explicitně tak, aby pokryla všechny možnosti tak, aby implicitní opětovné otevření nebylo požadováno. Uzavírání bylo zabráněno, protože fronta je lokální frontou typu definice MQQDT_TEMPORARY_DYNAMIC, která by byla zničena uzavřením.
MQRC_ATTRIBUTE_LOCKED (6104)	Byl proveden pokus o změnu hodnoty nebo atributu objektu, když je tento objekt otevřený. Určité atributy, jako např. **AlternateUserId**, nelze změnit, když je objekt otevřený.
MQRC_CURSOR_NOT_VALID (6105)	Kurzor procházení pro otevřenou frontu byl zneplatněn, protože byl naposledy použit implicitní znovuotevření. Nastavte volbu OpenOptions explicitně tak, aby pokryla všechny možnosti tak, aby implicitní opětovné otevření nebylo požadováno.
CHYBA MQRC_ENCODING_ERROR (6106)	Kódování další položky zprávy musí být MQENC_NATIVE pro čtení.
CHYBA MQRC_STRUTCID_ERROR (6107)	Struktura ID další položky zprávy, která je odvozena od 4 znaků začínajících na ukazateli dat, buď chybí, nebo je nekonzistentní s typem proměnné, do níž je položka čtena.
MQRC_NULL_POINTER (6108)	Byl zadán ukazatel s hodnotou Null, je-li požadován nebo odvozený ukazatel s hodnotou jinou než Null. Důvodem může být použití explicitních deklarací pro objekty WebSphere MQ použité z jazyka VBA jako parametrů pro volání (například dim msg jako objekt je v pořádku, ztlumená zpráva jako MqMessage může způsobit problémy). Například v aplikaci Excel s definovaným q a nastavením dim msg jako MqMessageq.put msg dává reasonCode MQRC_NULL_POINTER. Funkce funguje správně z adresáře VisualBasic.
ODKAZ MQRC_NO_CONNECTION_REFERENCE (6109)	Objekt **MQQueue** ztratil připojení k serveru **MQQueueManager**. K tomu dojde, pokud je příkaz **MQQueueManager**odpojen. Odstraňte objekt **MQQueue** .
MQRC_NO_BUFFER (6110)	Není k dispozici žádná vyrovnávací paměť. Pro objekt **MQMessage** nelze alokovat jednu vnitřní nekonzistenci ve stavu objektu, který by se neměl vyskytnout.
MQRC_BINARY_DATA_LENGTH_ERROR (6111), CHYBA	Délka binárních dat je nekonzistentní s délkou cílového atributu. Nula je správná délka pro všechny atributy. 24 je správná délka pro **CorrelationId** a pro **MessageId** 32 je správná délka pro **AccountingToken** .
MQRC_BUFFER_NOT_AUTOMATIC (6112)	Velikost uživatelsky definované a spravované vyrovnávací paměti nelze změnit. Vzhledem k tomu, že vyrovnávací paměť zpráv je spravována systémem, znamená to vnitřní nekonzistenci.
MQRC_INSUFFICIENT_BUFFER (6113)	Po ukazateli dat pro umístění požadavku není k dispozici dostatek prostoru vyrovnávací paměti. Důvodem může být skutečnost, že nelze změnit velikost vyrovnávací paměti.
NEDOSTATEČNÁ DATA MQRC_INSUFFICIENT_DATA (6114)	Po ukazateli dat pro uložení požadavku na čtení nejsou k dispozici dostatečná data. Zmenšete vyrovnávací paměť na správnou velikost a znovu si přečtěte data.
MQRC_DATA_OŘÍZNUTÁ (6115)	Data byla zkrácena při kopírování z jedné vyrovnávací paměti do jiné. Důvodem může být skutečnost, že nelze změnit velikost cílové vyrovnávací paměti, nebo protože se vyskytl problém s adresováním jedné nebo druhé vyrovnávací paměti, nebo protože vyrovnávací paměť je zmenšována s menší náhradou.

Tabulka 158. Kódy příčiny a o tom, co znamenají (pokračování)

Kód příčiny	Vysvětlení
HODNOTA MQRC_ZERO_LENGTH (6116)	Byla dodána nulová délka, kde je kladná délka buď povinná, nebo implikovaná.
MQRC_NEGATIVNÍ_DÉLKA (6117)	Byla zadána záporná délka, kde je vyžadována nulová nebo kladná délka.
MQRC_NEGATIVNÍ_POSUN (6118)	Byl zadán záporný posun, u kterého je vyžadována nulová nebo kladná odchylka.
FORMÁT MQRC_INCONSISTENT_FORMAT (6119)	Formát další položky zprávy je nekonzistentní s typem proměnné, do níž je položka čtena.
MQRC_INCONSISTENT_OBJECT_STATE (6120)	Existuje nekonzistence mezi tímto objektem, který je otevřený, a odkazovaným objektem MQQueueManager , který není připojen.
MQRC_CONTEXT_OBJECT_NOT_VALID (6121)	Odkaz na kontext voleb MQPutMessageneodkazuje na platný objekt MQQueue. Objekt byl již dříve zlikvidován.
CHYBA MQRC_CONTEXT_OPEN_ERROR (6122)	Odkaz na kontext voleb MQPutMessageodkazuje na objekt MQQueue, který nebylo možné otevřít pro vytvoření kontextu. Důvodem může být skutečnost, že objekt MQQueue má nevhodné volby otevření. Zkontrolujte kód příčiny odkazovaného objektu, abyste zavedli příčinu.
CHYBA MQRC_STRUC_LENGTH_ERROR (6123)	Délka vnitřní datové struktury je nekonzistentní s jejím obsahem. Pro MQRMH je délka nedostatečná k tomu, aby mohla obsahovat pevná pole a všechna data offsetu.
MQRC_NOT_CONNECTED (6124)	Metoda se nezdařila, protože nebylo k dispozici požadované připojení ke správci front a nebylo možné implicitně vytvořit připojení.
MQRC_NOT_OPEN (6125)	Metoda se nezdařila, protože objekt WebSphere MQ nebyl otevřen a otevření nemůže být provedeno implicitně.
MQRC_DISTRIBUTION_LIST_EMPTY (6126)	Otevření objektu MQDistributionList se nezdařilo, protože v seznamu distribucí nejsou žádné objekty položek MQDistributionList . Nápravná akce: Přidá alespoň jeden objekt položky MQDistributionListdo rozdělovníku.
MQRC_INCONSISTENT_OPEN_OPTIONS (6127)	Metoda selhala, protože objekt je otevřený a otevřené volby nejsou konzistentní s požadovanou operací. Nápravná akce: Otevřete objekt s vhodnými volbami otevření a zopakujte operaci.
MQRC_WRONG_VERSION (6128)	Metoda selhala, protože číslo verze zadané nebo zjištěné je buď chybné, nebo není podporované.

Nástroj úrovně kódu

Můžete být požádáni servisním týmem IBM o úroveň kódu, který jste nainstalovali.

Chcete-li to zjistit, spusťte obslužný program 'MQAXLEV'.

Z příkazového řádku přejděte do adresáře obsahujícího soubor MQAX200.dll nebo přidejte úplnou cestu a zadejte:

```
MQAXLev MQAX200.dll > MQAXLEV.OUT
```

kde MQAXLEV.OUT je název výstupního souboru.

Nezadáte-li výstupní soubor, zobrazí se na obrazovce podrobnosti.

Příklad výstupního souboru z nástroje na úrovni kódu je podrobně popsán v následujícím příkladu:

Příklad výstupního souboru z nástroje na úrovni kódu

```
5639-B43 (C) Copyright IBM Corp. 1996, 2024. ALL RIGHTS RESERVED.
     ***** Code Level is 5.1 ***** lib/mqole/mqole.cpp, mqole, p000, p000 L981119 1.8 98/08/21
lib/mqlsx/gmqdyn0a.c, mqlsx, p000, p000 L990212  1.6 99/02/11 16:40:24
lib/mqlsx/pc/gmqdyn1p.c, mqlsx, p000, p000 L990212  1.6 99/02/11 16:44:14
lib/mqlsx/xmqcsa.c, mqole, p000, p000 L990216             1.3 99/02/15 13:24:34
lib/mqlsx/xmqfdca.c, mqlsx, p000, p000 L990212               1.3 99/02/11 16:40:35
lib/mqlsx/xmqtrca.c, mqlsx, p000, p000 L990212             1.5 99/02/11 16:12:02
lib/mqlsx/xmqutila.c, mqlsx, p000, p000 L990212           1.3 99/02/11 16:40:40
lib/mqlsx/xmqutl1a.c, mqlsx, p000, p000 L990212           1.4 99/02/11 16:40:30
lib/mqlsx/xmqcnv1a.c, mqlsx, p000, p000 L990212           1.9 99/02/11 16:40:56
lib/mqlsx/xmqmsg.c, mqole, p000, p000 L990219             1.11 99/02/18 12:12:59
```

Rozhraní ActiveX k rozhraní MQAI

Stručný přehled rozhraní COM a jejich použití v rozhraní MQAI najdete v tématu “Použití rozhraní modelu objektu Component Interface (WebSphere MQ Automation Classes for ActiveX)” na stránce 1000.

Rozhraní MQAI umožňuje aplikacím vytvářet a odesílat příkazy PCF (Programmable Command Format) bez přímého získávání a formátování vyrovnávacích pamětí s proměnnou délkou požadovanou pro PCF. Další informace o rozhraní MQAI najdete v tématu Úvod do administrativního rozhraní produktu WebSphere MQ (MQAI). Třída MQAI ActiveX MQBag zapouzdřuje datové balíky podporované rozhraním MQAI způsobem, který je možné použít v libovolném jazyce, který podporuje vytváření objektů COM; například Visual Basic, C + +, Java a další skriptovací klienti ActiveX .

Rozhraní MQAI ActiveX je určen pro použití s třídami MQAX, které poskytují rozhraní COM pro rozhraní MQI. Další informace o třídách MQAX viz “Návrh aplikací MQAX, které přistupují k jiným aplikacím nežActiveX” na stránce 1001.

Rozhraní ActiveX poskytuje jednu třídu s názvem MQBag. Tato třída se používá k vytváření datových pytlů MQAI a jejich vlastnosti a metody jsou použity k vytvoření a práci s datovými položkami v každém balíku. Metoda Execute MQBag odesílá data balíku do správce front produktu WebSphere MQ jako zprávu PCF a shromažďuje odpovědi.

Další informace o třídě MQBag, jejích vlastnostech a metodách viz “Třída MQBag” na stránce 1076.

Zpráva PCF se odešle na uvedený objekt správce front, volitelně pomocí uvedených front požadavků a odpovědí. Odpovědi jsou vráceny v novém objektu MQBag. Úplná sada příkazů a odpovědí je popsána v části Definice formátů Programovatelných příkazů. Příkazy lze odesílat libovolnému správci front v síti produktu WebSphere MQ výběrem příslušných front požadavků a odpovědí.

Třída MQBag

Třída, MQBag, se používá k vytvoření objektů MQBag podle potřeby. Je-li vytvořena instance, třída MQBag vrátí nový odkaz na objekt MQBag.

Vytvořte objekt MQBag ve Visual Basic následujícím způsobem:

```
Dim mqbag As MQBag
Set mqbag = New MQBag
```

Vlastnost MQBag

Vlastnosti objektů MQBag jsou vysvětleny v následujícím seznamu:

- “Vlastnost položky” na stránce 1077.
- “Vlastnost Count” na stránce 1078.
- “Vlastnost Options” na stránce 1079.

Metody MQBag

Metody objektů MQBag jsou vysvětleny v následujícím seznamu:

- “Přidat metodu” na stránce 1079.
- “Metoda AddInquiry” na stránce 1080.
- “Vymazat metodu” na stránce 1080.
- “Provést metodu” na stránce 1081.
- “Metoda FromMessage” na stránce 1081.
- “Metoda ItemType” na stránce 1082.
- “metoda odebrání” na stránce 1082.
- “Metoda selektoru” na stránce 1083.
- “Metoda ToMessage” na stránce 1083.
- “Oříznout metodu” na stránce 1084.

Zpracování chyb

Pokud je při operaci na objektu MQBag zjištěna chyba, včetně chyb, které byly vráceny do balíku podkladovým objektem MQAX nebo MQAI, dojde k výjimce chyby. Třída MQBag podporuje informační rozhraní COM ISupportError, takže následující informace jsou k dispozici pro rutinu ošetření chyb:

- Číslo chyby: složený z kódu příčiny WebSphere MQ pro zjištěnou chybu a kód zařízení COM. Pole zařízení, jako standard pro COM, označuje oblast odpovědnosti za chybu. Pro chyby zjištěné produktem WebSphere MQ je to vždy FACILITY_ITF.
- Zdroj chyb: identifikuje typ a verzi objektu, který chybu zjistil. V případě chyb zjištěných během operací MQBag je tento zdroj chyb vždy MQBag.MQBag1.
- Popis chyby: řetězec poskytující symbolický název pro kód příčiny produktu WebSphere MQ .

Způsob přístupu k informacím o chybě závisí na jazyku skriptování. Například ve Visual Basicu se informace vrátí v objektu Err a v kódu příčiny WebSphere MQ je získán odečtením konstanty vbObjectz Err.Number.

ReasonCode = Err.Number -Chyba objektu vbObject

Pokud zpráva příkazu MQBag Execute odešle zprávu PCF a přijme se odpověď, operace se považuje za úspěšnou, ačkoli selhal příkaz, který byl odeslán. V tomto případě obsahuje samotný vak pro odpověď kódy příčiny dokončení a chyby, jak je popsáno v tématu Definice formátů Programovatelných příkazů .

Vlastnost položky

Účel

Vlastnost Položka reprezentuje položku v balíku. Používá se k nastavení nebo dotazu na hodnotu položky. Použití této vlastnosti odpovídá následujícím voláním MQAI:

- "ŘetězecmqSet"
- "mqSetCelé číslo"
- "mqInquireCelé číslo"
- "ŘetězecmqInquire"
- "mqInquireBag"

v příručce Odkaz na formáty PCF (Programmable Command Format).

Formát

Položka (Selektor, ItemIndex, Hodnota)

Parametry

***Selector* (VARIANT)-vstup**
Selektor položky, která má být nastavena, nebo vyšetřován.

Při zjišťování informací o položce je výchozím nastavením hodnota MQSEL_ANY_USER_SELECTOR. Při nastavování položky je výchozím nastavením hodnota MQIA_LIST nebo MQCA_LIST.

Pokud `Selector` není typu long, hodnota MQRC_SELECTOR_TYPE_ERROR.

Tento parametr je volitelný.

***ItemIndex* (LONG)-vstup**
Tato hodnota identifikuje výskyt položky uvedeného selektoru, který má být nastaven nebo se má provést zjišťování. MQIND_NONE je výchozí hodnota.

Tento parametr je volitelný.

***Value* (VARIANT)-vstupní/výstupní**
Vrácená hodnota nebo hodnota, která má být nastavena. Při zjišťování informací o položce může být návratová hodnota typu long, string nebo MQBag. Když však nastavujete položku, musí být hodnota typu long nebo string; pokud tomu tak není, výsledky MQRC_ITEM_VALUE_ERROR.

Vyvolání jazyka Visual Basic

Při zjišťování informací o hodnotě položky v rámci balíku:

```
Value = mqbag[.Item]([Selector],
[ItemIndex])
```

Pro odkazy MQBag:

```
Set abag = mqbag[.Item]([Selector].
[ItemIndex])
```

Chcete-li nastavit hodnotu položky v balíku, postupujte takto:

```
mqbag[.Item]([Selector],
[ItemIndex]) = Value
```

Vlastnost Count

Účel

Vlastnost Count představuje počet datových položek v balíku. Tato vlastnost odpovídá volání MQAI, "mqCountItems," v příručce Odkaz na formáty PCF (Programmable Command Format).

Formát

Počet (*Selector, Value*)

Parametry

***Selector* (VARIANT)-vstup**
Selektor datových položek, které mají být zahrnuty do počtu.

MQSEL_ALL_USER_SELECTORS je výchozí hodnota.

Pokud `Selector` není typu long, je vrácen objekt MQRC_SELECTOR_TYPE_ERROR.

***Value* (LONG)-výstup**
Počet položek v balíku zahrnutých do *Selector*.

Vyvolání jazyka Visual Basic

Chcete-li vrátit počet položek v balíku:

```
ItemCount = mqbag.Count([Selector])
```

Vlastnost Options

Účel

Volby vlastnosti Volby nastavuje volby pro použití balíku. Tato vlastnost odpovídá parametru Options volání MQAI, "mqCreateBag," v publikaci Odkaz na formáty PCF (Programmable Command Format).

Formát

Volby (*Options*)

Parametry

***Options* (LONG)-vstupní/výstupní**
Volby balíku.

Poznámka: Volby balíku musí být nastaveny ***před*** datovými položkami, aby byly přidány nebo nastaveny v rámci balíku. Pokud jsou volby změněny, když balík není prázdný, výsledky chyb MQRC_OPTIONS_ERROR. To platí i v případě, že se obal následně vymaže.

Vyvolání jazyka Visual Basic

Při zjišťování informací o volbách položky v rámci balíku:

```
Options = mqbag.Options
```

Chcete-li nastavit volbu položky v balíku, postupujte takto:

```
mqbag.Options = Options
```

Metody MQBag

Metody objektů MQBag jsou vysvětleny na následujících stránkách.

Přidat metodu

Účel

Metoda Add přidá datovou položku do balíku. Tato metoda odpovídá volání MQAI, "mqAddInteger" a "mqAddString," v příručce Odkaz na formáty PCF (Programmable Command Format).

Formát

Přidat (*Value, Selector*)

Parametry

***Value* (VARIANT)-vstup**
Celé číslo nebo řetězcová hodnota datové položky.

***Selector* (VARIANT)-vstup**
Selektor identifikující položku, která má být přidána.

V závislosti na typu `Value`je výchozí hodnota MQIA_LIST nebo MQCA_LIST. Pokud parametr `Selector` není typu long, výsledky MQRC_SELECTOR_TYPE_ERROR.

Vyvolání jazyka Visual Basic

Chcete-li přidat položku do balíku:

```
mqbag.Add(Value,[Selector])
```

Metoda AddInquiry

Účel

Metoda AddInquiry přidá selektor uvádějící atribut, který se má vrátit, když se odešle balík administrace k provedení příkazu INQUIRE. Tato metoda odpovídá volání MQAI, "mqAddInquiry,", v produktu Odkaz na formáty PCF (Programmable Command Format).

Formát

AddInquiry *(Inquiry)*

Parametry

***Inquiry* (LONG)-vstup**
Selektor atributu produktu WebSphere MQ , který má být vrácen administrativním příkazem INQUIRE.

Vyvolání jazyka Visual Basic

Chcete-li použít metodu AddInquiry :

```
mqbag.AddInquiry(Inquiry)
```

Vymazat metodu

Účel

Metoda Clear odstraní všechny datové položky z balíku. Tato metoda odpovídá volání MQAI, "mqClearBag,", v rámci Odkaz na formáty PCF (Programmable Command Format).

Formát

Vymazat

Vyvolání jazyka Visual Basic

Chcete-li odstranit všechna data itmes z balíku:

```
mqbag.Clear
```

Provést metodu

Účel

Metoda Execute odešle na příkazový server zprávu příkazu administrace a čeká na všechny zprávy odpovědí. Tato metoda odpovídá volání MQAI, "mqExecute,", v Odkaz na formáty PCF (Programmable Command Format).

Formát

Spuštění produktu *(QueueManager, Command, OptionsBag, RequestQ, ReplyQ, ReplyBag*)

Parametry

***QueueManager* (MQQueueManager)-vstup**
Správce front, ke kterému je aplikace připojena.

***Command* (LONG)-vstup**
Příkaz, který má být proveden.

***OptionsBag* (MQBag)-vstup**
Sáček obsahující volby, které ovlivňují zpracování volání.

***RequestQ* (MQQueue)-vstup**
Fronta, do které bude umístěna zpráva příkazu administrace.

***ReplyQ* (MQQueue)-vstup**
Fronta, na které jsou přijímány všechny zprávy odpovědi.

***ReplyBag* (MQBag)-výstup**
Odkaz na balík obsahující data ze zpráv odpovědi.

Vyvolání jazyka Visual Basic

Chcete-li odeslat zprávu s příkazovým příkazem a čekat na všechny zprávy odpovědí, postupujte takto:

```
Set ReplyBag = mqbag.Execute(QueueManager, Command,
[OptionsBag],[RequestQ],[ReplyQ])
```

Metoda FromMessage

Účel

Metoda FromMessage načte data ze zprávy do tašky. Tato metoda odpovídá volání MQAI, "mqBufferToBag," v rámci Odkaz na formáty PCF (Programmable Command Format).

Formát

FromMessage *(Message, OptionsBag*)

Parametry

***Message* (MQMessage)-vstup**
Zpráva obsahující data, která mají být převedena.

***OptionsBag* (MQBag)-vstup**
Volby pro řízení zpracování volání.

Vyvolání jazyka Visual Basic

Chcete-li načíst data ze zprávy do balíku, postupujte takto:

```
mqbag.FromMessage(Message,[OptionsBag])
```

Metoda ItemType

Účel

Metoda ItemType vrací typ hodnoty v zadané položce v balíku. Tato metoda odpovídá volání MQAI, "mqInquireItemInfo," v publikaci Odkaz na formáty PCF (Programmable Command Format).

Formát

ItemType *(Selector, ItemIndex, ItemType*)

Parametry

***Selector* (VARIANT)-vstup**
Selektor identifikující položku, která má být dotazovaná.

MQSEL_ANY_USER_SELECTOR je výchozí hodnota. Pokud parametr `Selector` není typu long, výsledky MQRC_SELECTOR_TYPE_ERROR.

***ItemIndex* (LONG)-vstup**
Index položek, které mají být dotazovány.

MQIND_NONE je výchozí hodnota.

***ItemType* (LONG)-výstup**
Datový typ zadané položky.

Poznámka: Musí být zadán buď parametr `Selector` , `ItemIndex` nebo oba parametry. Není-li zadán žádný parametr, budou výsledkem operace MQRC_PARAMETER_MISSING.

Vyvolání jazyka Visual Basic

Chcete-li vrátit typ hodnoty, postupujte takto:

```
ItemType = mqbag.ItemType([Selector],
[ItemIndex])
```

metoda odebrání

Účel

Metoda Odebrat odstraní položku z balíku. Tato metoda odpovídá volání MQAI, "mqDeleteItem," v Odkaz na formáty PCF (Programmable Command Format).

Formát

Odebrat *(Selector, ItemIndex*)

Parametry

***Selector* (VARIANT)-vstup**
Selektor identifikující položku, která má být odstraněna.

MQSEL_ANY_USER_SELECTOR je výchozí hodnota. Pokud parametr `Selector` není typu long, výsledky MQRC_SELECTOR_TYPE_ERROR.

***ItemIndex* (LONG)-vstup**
Index položky, která má být odstraněna.

MQIND_NONE je výchozí hodnota.

Poznámka: Musí být zadán buď parametr `Selector` , `ItemIndex` nebo oba parametry. Není-li zadán žádný parametr, budou výsledkem operace MQRC_PARAMETER_MISSING.

Vyvolání jazyka Visual Basic

Chcete-li odstranit položku z balíku:

```
mqbag.Remove([Selector],[ItemIndex])
```

Metoda selektoru

Účel

Metoda Selector vrací selektor zadané položky v rámci balíku. Tato metoda odpovídá volání MQAI, "mqInquireItemInfo," v rámci Odkaz na formáty PCF (Programmable Command Format).

Formát

Selektor ***(Selector, ItemIndex, OutSelector)***

Parametry

Selector **(VARIANT)-vstup**
Selektor identifikující položku, která má být dotazovaná.

MQSEL_ANY_USER_SELECTOR je výchozí hodnota. Pokud parametr `Selector` není typu long, výsledky MQRC_SELECTOR_TYPE_ERROR.

ItemIndex **(LONG)-vstup**
Index položky, která má být dotazovaná.

MQIND_NONE je výchozí hodnota.

OutSelector **(VARIANT)-výstup**
Selektor zadané položky.

Poznámka: Musí být zadán buď parametr `Selector` , `ItemIndex` nebo oba parametry. Není-li zadán žádný parametr, budou výsledkem operace MQRC_PARAMETER_MISSING.

Vyvolání jazyka Visual Basic

Chcete-li vrátit selektor položky, postupujte takto:

```
OutSelector = mqbag.Selector([Selector],
[ItemIndex])
```

Metoda ToMessage

Účel

Metoda ToMessage vrací odkaz na objekt MQMessage. Odkaz obsahuje data z balíku. Tato metoda odpovídá volání MQAI, "mqBagToBuffer," v publikaci Odkaz na formáty PCF (Programmable Command Format).

Formát

ToMessage *(OptionsBag, Message)*

Parametry

***OptionsBag* (MQBag)-vstup**
Sáček obsahující volby, které řídí zpracování metody.

***Message* (MQMessage)-výstup**
Odkaz na objekt MQMessage obsahující data z balíku.

Vyvolání jazyka Visual Basic

Chcete-li použít metodu ToMessage , postupujte takto:

```
Set Message = mqbag.ToMessage([OptionsBag])
```

Oříznout metodu

Účel

Metoda Truncate snižuje počet uživatelských položek v balíku. Tato metoda odpovídá volání MQAI, "mqTruncateBag," v příručce Odkaz na formáty PCF (Programmable Command Format).

Formát

Oseknout *(ItemCount)*

Parametry

***ItemCount* (LONG)-vstup**
Počet položek uživatele, které mají zůstat v balíku po oříznutí.

Vyvolání jazyka Visual Basic

Chcete-li snížit počet uživatelských položek v balíku, postupujte takto:

```
mqbag.Truncate(ItemCount)
```

O ukázkách produktu WebSphere MQ Automation Classes for ActiveX

Tato příloha popisuje ukázky WebSphere MQ Automation Classes for ActiveX Starter samples a vysvětluje, jak je používat.

Produkt WebSphere MQ for Windows poskytuje následující ukázkové programy Visual Basic:

- MQAXTRIV.VBP
- MQAXBSRV.VBP
- MQAXDLST.VBP
- MQAXCLSS.VBP

Tyto ukázky jsou spuštěny ve Visual Basic 4 nebo Visual Basic 5. Naleznete je v adresáři ... \tools\mqax\samples\vb.

Ve stejném adresáři také najdete ukázky pro Microsoft Excel a html. Patří mezi ně:

- MQAX.XLS
- MQAXTRIV.XLS
- MQAXTRIV.HTM

Poznámka: Používáte-li Visual Basic 5, **musíte** vybrat a instalovat komponentu Visual Basic grid32.ocx.

Co je prokázáno ve vzorcích

Ukázky demonstrují použití tříd automatizace produktu WebSphere MQ pro ActiveX pro:

- Připojit se ke správci front
- Přístup k frontě
- Vložit zprávu do fronty
- Získat zprávu z fronty

Centrální část vzorku Visual Basic se zobrazí na následujících stránkách.

"Příprava na spuštění ukázek" na stránce 1085 a

"Ošetření chyb ve vzorcích" na stránce 1086

Spuštění ukázek ActiveX Starter

Před spuštěním produktu WebSphere MQ Automation Classes for ActiveX Starter samples zkontrolujte, zda je spuštěn výchozí správce front a že jste vytvořili požadované definice front. Podrobnosti o vytvoření a spuštění správce front a vytvoření fronty naleznete v tématu Administrace. Ukázka používá frontu SYSTEM.DEFAULT.LOCAL.QUEUE , která by měla být definována na kterémkoli standardním nastavení serveru WebSphere MQ .

Různé způsoby použití datových pytlů jsou uvedeny v následujícím seznamu:

- Připojit se ke správci front
- Přístup k frontě
- Vložit zprávu do fronty
- Získat zprávu z fronty

Informace o ukázkách spouštěče MQAX pro produkt Microsoft Basic verze 4 nebo novější naleznete v tématu "Spuštění ukázky MQAXTRIV" na stránce 1086 .

Informace o ukázce, která umožňují procházet vlastnosti a metody správců front a objektů front, naleznete v tématu "Spuštění ukázky MQAXCLSS" na stránce 1087 .

Informace o ukázce MQAXDLST obsahuje "Ukázka MQAXDLST" na stránce 1088

Informace o spuštění ukázky struktury MQAX pro produkt Microsoft Excel 95 nebo novější verze MQAXTRIV.XLS, viz "Spuštění příkazu MQAXTRIV.XLS" na stránce 1088.

Informace o spuštění ukázkové aplikace Bank s produktem MQAX.XLS, viz "Spuštění ukázkové aplikace Bank s parametrem MQAX.XLS" na stránce 1088

Informace o startovní ukázce pomocí kompatibilního webového prohlížeče ActiveX viz "Ukázkový příklad použití kompatibilního webového prohlížeče ActiveX" na stránce 1088

Příprava na spuštění ukázek

Chcete-li spustit některý z ukázek, je třeba provést jednu z následujících možností v závislosti na tom, které ukázky hodláte spustit.

- Microsoft Visual Basic verze 4 (nebo novější)
- Microsoft Excel 95 (nebo novější)
- Webový prohlížeč

Potřebujete také:

- Správce front produktu WebSphere MQ je spuštěn.
- Fronta produktu WebSphere MQ je již definována.

Ošetření chyb ve vzorcích

Většina ukázek poskytnutých v balíku WebSphere MQ Automation Classes for ActiveX vykazuje malou nebo žádnou chybu při zpracování chyb. Další informace o ošetření chyb viz “Ošetření chyb” na stránce 1005.

Spuštění ukázky MQAXTRIV

1. Spusťte správce front.
2. V Průzkumníku Windows nebo File Managervyberte ikonu ukázky MQAXTRIV.VBP (soubor projektu Visual Basic) a otevřete soubor.

   Spustí se program Visual Basic a otevře se soubor MQAXTRIV.VBP.
3. V Visual Basic stiskněte funkční klávesu 5 (F5) pro spuštění ukázky.
4. Klepněte kdekoli ve formuláři okna, "Triviální tester MQAX".

Pokud vše funguje správně, pozadí okna by se mělo změnit na zelenou. Pokud se vyskytl problém s nastavením, pozadí okna by se mělo změnit na červené a chybové informace se zobrazí.

Následující obrázek ukazuje centrální část ukázky jazyka Visual Basic.

```
Option Explicit

Private Sub Form_Click()

'*****************************************************************************
'* This simple example illustrates how to put and get a WebSphere MQ message to
'* and from a WebSphere MQ message queue. The data from the message returned by the
'*get is read and compared with that from the original message.
'*****************************************************************************
Dim MQSess As MQSession                 '* session object
Dim QMgr As MQQueueManager              '* queue manager object
Dim Queue As MQQueue                    '* queue object
Dim PutMsg As MQMessage                 '* message object for put
Dim GetMsg As MQMessage                 '* message object for get
Dim PutOptions As MQPutMessageOptions   '* get message option

Dim GetOptions As MQGetMessageOptions   '* put message options
Dim PutMsgStr As String                 '* put message data string
Dim GetMsgStr As String                 '* get message data string
'*****************************************************************************
'* Handle errors
'*****************************************************************************
On Error GoTo HandleError

'*****************************************************************************
'* Initialize the current position for the form
'*****************************************************************************
CurrentX = 0
CurrentY = 0

'*****************************************************************************
'* Create the MQSession object and access the MQQueueManager and (local) MQQueue
'*****************************************************************************
Set MQSess = New MQSession
Set QMgr = MQSess.AccessQueueManager("")
Set Queue = QMgr.AccessQueue("SYSTEM.DEFAULT.LOCAL.QUEUE", _
                        MQOO_OUTPUT Or MQOO_INPUT_AS_Q_DEF)

'*****************************************************************************
'* Create a new MQMessage object for use with put, add some data then create an
'* MQPutMessageOptions object and put the message
'*****************************************************************************
Set PutMsg = MQSess.AccessMessage()
PutMsgStr = "12345678 " & Time
PutMsg.MessageData = PutMsgStr
Set PutOptions = MQSess.AccessPutMessageOptions()
Queue.Put PutMsg, PutOptions

'*****************************************************************************
'* Create a new MQMessage object for use with get, set the MessageId (to that of
'* the message that was put), create an MQGetMessageOptions object and get the
```

```
'* message.
'*
'* Note: Setting the MessageId ensures that the get returns the MQMessage
'* that was put earlier.
'*******************************************************************************

Set GetMsg = MQSess.AccessMessage()
GetMsg.MessageId = PutMsg.MessageId
Set GetOptions = MQSess.AccessGetMessageOptions()
Queue.Get GetMsg, GetOptions
'*******************************************************************************
'* Read the data from the message returned by the get, compare it with
'* that from the original message and output a suitable message.
'*******************************************************************************
GetMsgStr = GetMsg.MessageData
Cls
If GetMsgStr = PutMsgStr Then
    BackColor = RGB(127, 255, 127)  '* set to green for ok
    Print
    Print "Message data comparison was successful."
    Print "Message data: """ & GetMsgStr & """"

Else
    BackColor = RGB(255, 255, 127)  '* set to amber for compare error
    Print "Compare error: "
    Print "The message data returned by the get did not match the " &_
    "input data from the original message that was put."
    Print
    Print "Input message data:        """ & PutMsgStr & """"
    Print "Returned message data: """ & GetMsgStr & """"
End If

Exit Sub
'*******************************************************************************
'* Handle errors
'*******************************************************************************
HandleError:
Dim ErrMsg As String
Dim StrPos As Integer

Cls
BackColor = RGB(255, 0, 0)  '* set to red for error
Print "An error occurred as follows:"
Print ""
If MQSess.CompletionCode <> MQCC_OK Then
    ErrMsg = Err.Description
    StrPos = InStr(ErrMsg, " ")            '* search for first blank
    If StrPos > 0 Then
        Print Left(ErrMsg, StrPos)         '* print offending MQAX object name
    Else
        Print Error(Err)                   '* print complete error object
    End If
    Print ""
    Print "WebSphere MQ Completion Code = " & MQSess.CompletionCode
    Print "WebSphere MQ Reason Code = " & MQSess.ReasonCode
    Print "(" & MQSess.ReasonName & ")"
Else
    Print "Visual Basic error:  " & Err
    Print Error(Err)
End If

Exit Sub

End Sub
```

Spuštění ukázky MQAXCLSS

Tato ukázka vám umožňuje procházet vlastnosti a metody správců front a objektů front.

1. Spusťte správce front.
2. Otevřete soubor MQAXCLSS.VBP, poklepáním na ikonu dokumentu v Průzkumníku Windows nebo klepnutím na Soubor-Otevřít z nabídky Soubor ve Visual Basic.
3. Spusťte ukázku.
4. Zadejte příslušné správce front a názvy front a poté klepněte na příslušná tlačítka.

Ukázka MQAXDLST

Ukázka kódu Visual Basic MQAXDLST demonstruje použití distribučního seznamu k odeslání stejné zprávy do dvou front s jedním vkládanou. Chcete-li ukázku spustit, proveďte to samé jako pro ukázku MQAXCLSS.

Ukázka MQAX Starter pro produkt Microsoft Excel 95 nebo novější

Tento oddíl vysvětluje, jak spustit ukázku spouštěče MQAX pro produkt Microsoft Excel 95 nebo novější, MQAXTRIV.XLS.

Spuštění příkazu MQAXTRIV.XLS

1. Spusťte správce front.
2. V Průzkumníku nebo File Managervyberte ikonu pro vzorek MQAX MQAXTRIV.XLS.
3. Klepněte na tlačítko v sešitu.
4. Obrazovka je aktualizována zprávou o úspěchu (nebo selhání).

Spuštění ukázkové aplikace Bank s parametrem MQAX.XLS

Chcete-li spustit demonstraci Bank, postupujte podle následujících kroků.

1. Spusťte správce front.
2. Spusťte příkazový soubor IBM WebSphere MQ MQSC, BANK.TST. Tím se nastaví potřebné definice front produktu IBM WebSphere MQ .

   Chcete-li zjistit, jak používat příkazový soubor MQSC, prostudujte si téma Příkazy skriptu MQSC (Script).
3. Spusťte příkaz MQAXBSRV.VBP. Tento ukázkový program je server simulující back-endovou aplikaci a musí být spuštěn s aplikací Microsoft Excel.
4. Spusťte MQAX.XLS. Tato ukázka je ukázkou klienta IBM WebSphere MQ .
5. Vyberte zákazníka ze seznamu.
6. Klepněte na tlačítko **Odeslat**.

Po krátké přestávce (přibližně 3 sekundy) se pole naplní hodnotami a zobrazí se sloupcový graf.

Ukázkový příklad použití kompatibilního webového prohlížeče ActiveX

Poznámka: Chcete-li spustit tuto ukázku, musíte spustit webový prohlížeč kompatibilní s ActiveX . Prohlížeč Microsoft Internet Explorer (ale ne Netscape Navigator) je kompatibilní webový prohlížeč.

Spuštění ukázky HTML

Tato ukázka demonstruje, jak lze vyvolat MQAX z obou jazyků VBScript i jazyka JavaScript.

1. Spusťte správce front.
2. Otevřete soubor, "MQAXTRIV.HTM", ve webovém prohlížeči kompatibilním s ActiveX .

   Můžete to provést buď dvojitým klepnutím na ikonu souboru v Průzkumníku Windows , nebo můžete zvolit Soubor-Otevřít z nabídky Soubor ve webovém prohlížeči kompatibilního s ActiveX .
3. Postupujte podle pokynů na obrazovce.

Poznámky

Tyto informace byly vyvinuty pro produkty a služby poskytované v USA.

Společnost IBM nemusí nabízet produkty, služby nebo funkce uvedené v tomto dokumentu v jiných zemích. Informace o produktech a službách, které jsou ve vaší oblasti aktuálně dostupné, získáte od místního zástupce společnosti IBM. Odkazy na produkty, programy nebo služby společnosti IBM v této publikaci nejsou míněny jako vyjádření nutnosti použití pouze uvedených produktů, programů či služeb společnosti IBM. Místo toho lze použít jakýkoli funkčně ekvivalentní produkt, program nebo službu, které neporušují žádná práva k duševnímu vlastnictví IBM. Ověření funkčnosti produktu, programu nebo služby pocházející od jiného výrobce je však povinností uživatele.

Společnost IBM může vlastnit patenty nebo nevyřízené žádosti o patenty zahrnující předměty popsané v tomto dokumentu. Vlastnictví tohoto dokumentu neposkytuje licenci k těmto patentům. Dotazy týkající se licencí můžete posílat písemně na adresu:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Odpovědi na dotazy týkající se licencí pro dvoubajtové znakové sady (DBCS) získáte od oddělení IBM Intellectual Property Department ve vaší zemi, nebo tyto dotazy můžete zasílat písemně na adresu:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

Následující odstavec se netýká Velké Británie nebo kterékoliv jiné země, kde taková opatření odporují místním zákonům: SPOLEČNOST INTERNATIONAL BUSINESS MACHINES CORPORATION TUTO PUBLIKACI POSKYTUJE TAKOVOU, "JAKÁ JE" , BEZ JAKÝCHKOLIV ZÁRUK, VYJÁDŘENÝCH VÝSLOVNĚ NEBO VYPLÝVAJÍCÍCH Z OKOLNOSTÍ, VČETNĚ, A TO ZEJMÉNA, ZÁRUK NEPORUŠENÍ PRÁV TŘETÍCH STRAN, PRODEJNOSTI NEBO VHODNOSTI PRO URČITÝ ÚČEL VYPLÝVAJÍCÍCH Z OKOLNOSTÍ. Některé právní řády u určitých transakcí nepřipouštějí vyloučení záruk výslovně vyjádřených nebo vyplývajících z okolností, a proto se na vás toto omezení nemusí vztahovat.

Uvedené údaje mohou obsahovat technické nepřesnosti nebo typografické chyby. Údaje zde uvedené jsou pravidelně upravovány a tyto změny budou zahrnuty v nových vydáních této publikace. Společnost IBM může kdykoli bez upozornění provádět vylepšení nebo změny v produktech či programech popsaných v této publikaci.

Veškeré uvedené odkazy na webové stránky, které nespravuje společnost IBM, jsou uváděny pouze pro referenci a v žádném případě neslouží jako záruka funkčnosti těchto webů. Materiály uvedené na tomto webu nejsou součástí materiálů pro tento produkt IBM a použití uvedených stránek je pouze na vlastní nebezpečí.

Společnost IBM může použít nebo distribuovat jakékoli informace, které jí sdělíte, libovolným způsobem, který společnost považuje za odpovídající, bez vyžádání vašeho svolení.

Vlastníci licence k tomuto programu, kteří chtějí získat informace o možnostech (i) výměny informací s nezávisle vytvořenými programy a jinými programy (včetně tohoto) a (ii) oboustranného využití vyměňovaných informací, mohou kontaktovat informační středisko na adrese:

IBM Corporation
Koordinátor spolupráce softwaru, oddělení 49XA
148 00 Praha 4-Chodby

148 00 Praha 4-Chodov
U.S.A.

Poskytnutí takových informací může být podmíněno dodržením určitých podmínek a požadavků zahrnujících v některých případech uhrazení stanoveného poplatku.

IBM poskytuje licencovaný program popsaný v těchto informacích a veškeré dostupné licencované materiály na základě podmínek smlouvy IBM Customer Agreement, IBM International Program License Agreement nebo jiné ekvivalentní smlouvy mezi námi.

Jakékoli údaje o výkonnosti obsažené v této publikaci byly zjištěny v řízeném prostředí. Výsledky získané v jakémkoli jiném operačním prostředí se proto mohou výrazně lišit. Některá měření mohla být prováděna na vývojových verzích systémů a není zaručeno, že tato měření budou stejná i na běžně dostupných systémech. Některá měření mohla být navíc odhadnuta pomocí extrapolace. Skutečné výsledky mohou být jiné. Čtenáři tohoto dokumentu by měli zjistit použitelné údaje pro své specifické prostředí.

Informace týkající se produktů jiných výrobců pocházejí od dodavatelů těchto produktů, z jejich veřejných oznámení nebo z jiných veřejně dostupných zdrojů. Společnost IBM tyto produkty netestovala a nemůže potvrdit správný výkon, kompatibilitu ani žádné jiné výroky týkající se produktů jiných výrobců než IBM. Otázky týkající se kompatibility produktů jiných výrobců by měly být směrovány dodavatelům těchto produktů.

Veškerá tvrzení týkající se budoucího směru vývoje nebo záměrů společnosti IBM se mohou bez upozornění změnit nebo mohou být zrušena a reprezentují pouze cíle a plány společnosti.

Tyto údaje obsahují příklady dat a sestav používaných v běžných obchodních operacích. Aby byla představa úplná, používají se v příkladech jména osob a názvy společností, značek a produktů. Všechna tato jména a názvy jsou fiktivní a jejich podobnost se jmény, názvy a adresami používanými ve skutečnosti je zcela náhodná.

LICENČNÍ INFORMACE:

Tyto informace obsahují ukázkové aplikační programy ve zdrojovém jazyce ilustrující programovací techniky na různých operačních platformách. Tyto ukázkové programy můžete bez závazků vůči společnosti IBM jakýmkoli způsobem kopírovat, měnit a distribuovat za účelem vývoje, používání, odbytu či distribuce aplikačních programů odpovídajících rozhraní API pro operační platformu, pro kterou byly ukázkové programy napsány. Tyto příklady nebyly plně testovány za všech podmínek. Společnost IBM proto nemůže zaručit spolehlivost, upotřebitelnost nebo funkčnost těchto programů.

Při prohlížení těchto dokumentů v elektronické podobě se nemusí zobrazit všechny fotografie a barevné ilustrace.

Informace o programovacím rozhraní

Informace programátorských rozhraní, je-li poskytnuta, vám pomohou vytvořit aplikační software pro použití s tímto programem.

Tato příručka obsahuje informace o zamýšlených programovacích rozhraních, které umožňují zákazníkům psát programy za účelem získání služeb produktu IBM WebSphere MQ.

Tyto informace však mohou obsahovat i diagnostické údaje a informace o úpravách a ladění. Informace o diagnostice, úpravách a vyladění jsou poskytovány jako podpora ladění softwarových aplikací.

Důležité: Nepoužívejte tyto informace o diagnostice, úpravách a ladění jako programátorské rozhraní, protože se mohou měnit.

Ochranné známky

IBM, logo IBM , ibm.comjsou ochranné známky společnosti IBM Corporation, registrované v mnoha jurisdikcích po celém světě. Aktuální seznam ochranných známek IBM je k dispozici na webu na stránce "Copyright and trademark information"www.ibm.com/legal/copytrade.shtml. Ostatní názvy produktů a služeb mohou být ochrannými známkami společnosti IBM nebo jiných společností.

Microsoft a Windows jsou ochranné známky společnosti Microsoft Corporation ve Spojených státech a případně v dalších jiných zemích.

UNIX je registrovaná ochranná známka skupiny The Open Group ve Spojených státech a případně v dalších jiných zemích.

Linux je registrovaná ochranná známka Linuse Torvaldse ve Spojených státech a případně v dalších jiných zemích.

Tento produkt obsahuje software vyvinutý v rámci projektu Eclipse Project (http://www.eclipse.org/).

Java a všechny ochranné známky a loga založené na termínu Java jsou ochranné známky nebo registrované ochranné známky společnosti Oracle anebo příbuzných společností.

Číslo položky:

(1P) P/N: